4601
640.02
230-1

NBSIR 74-570 (R)

Standards Analysis for Future WWMCCS Computer Networking

Dennis W. Fife. Editor

Systems and Software Division
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington. D. C. 20234

August 30, 1974

Final Report

Prepared for
Joint Technical Support Activity
Defense Communications Agency
Washington, D. C. 20305

NBSIR 74-570

STANDARDS ANALYSIS FOR FUTURE WWMCCS COMPUTER NETWORKING

Dennis W. Fife. Editor

Systems and Software Division
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D. C. 20234

August 30, 1974

Final Report

Prepared for
Joint Technical Support Activity
Defense Communications Agency
Washington, D. C. 20305

U. S. DEPARTMENT OF COMMERCE, Frederick B. Dent, Secretary

NATIONAL BUREAU OF STANDARDS, Richard W. Roberts, Director

TABLE OF CONTENTS

FOREWORD

This report is submitted to Mr. Jack Kroboth of the Standards Branch, Joint Technical Support Activity, in fulfillment of an interagency agreement between the National Bureau of Standards and the Defense Communications Agency. In addition to the editor, principal contributors from the Institute for Computer Sciences and Technology were: Jules Aronson, Gloria Bolotsky, Ira Cotton, Frances E. Holberton, Elizabeth Parker, Zella Ruthberg. Brian Lucas also assisted through consulting and review of the material.

STANDARDS ANALYSIS FOR FUTURE WWMCCS COMPUTER NETWORKING

ABSTRACT

The World-Wide Military Command and Control System (WWMCCS) comprises over 30 computer installations, evolving toward heavy dependence on terminal-to-computer and computer-to-computer communications in serving individual bases, commands, and national military authorities. Although a highly standardized system has emerged through identical mainframes, hardware devices, and basic software, the hardware/software configurations of individual installations are not identical and significant evolutionary changes are occurring in hardware/software components, interfaces, and configuration rules. It is vital therefore to deliberately adopt selected interface specifications as standards that transcend the present computer hardware and software, and that will guide near-term reconfiguration or redesign as well as ultimate, progressive replacement of the WWMCCS computer network system. A previous NBS report, Technical Note 843, provided a succinct handbook on existing computer-communications standards with widespread applicability. This report addresses issues and methods in interface standards analysis, to identify and partially evaluate computer-communications techniques that would contribute to enhancing WWMCCS interoperability. These issues include: functional decomposition of computer-communications processes; complexity and performance analysis of advanced data link control techniques; feasibility of common communications front-end software; and requirements for user-oriented network protocols. The presentation indicates the standard "discipline" involved in the evaluation and implementation of effective standards for a complex computer network environment.

1. INTRODUCTION

1.1 Background

The World-Wide Military Command and Control Systems (WWMCCS), comprising over 30 installations of Honeywell 6000 computer hardware, is evolving toward heavy dependence on terminal-to-computer and computer-to-computer communications in providing service to individual bases, commands, and national military authorities. From one viewpoint, the use of identical mainframes, hardware devices, and common software, coupled with centralized technical assistance and management, leads to a highly standardized system. Thus, the term "WWMCCS Standard System" is used in reference to the Honeywell 6000 hardware/software configuration. On the other hand, the hardware/software configuration of any one installation is not unchanging and furthermore, significant evolutionary changes are being made in the common hardware/software components, interfaces, and configuration rules. Such changes impact existing and future computer - communications operations as well as strictly data processing applications. Without continuing effort to stabilize and manage certain interface* specifications and configuration criteria, the benefits of selective initial standardization may erode in a plethora of local options and secondary modifications, resulting in degraded reliability and interoperability, and forestalling low-cost replacement and upgrade of system parts. It is vital then to undertake the deliberate adoption of selected interface specifications as standards that transcend the present hardware/software, to guide near-term reconfiguration, evolutionary redesign, and ultimate, progressive replacement of the overall computer network system.

*Throughout this report, the term "interfaces" is used in the general sense, as affecting hardware, software, and system users.

1.2 Purpose of This Report

Establishing long-term standards for computer network interfaces begins with the recognition and application of pertinent existing standards. A previous effort by NBS[1.1] developed a succinct handbook on existing computer-communications standards with widespread applicability. This report addresses technical issues and methods in interface standards analysis, to identify and partially evaluate meaningful computer-communications processing parameters and procedures that would contribute to enhancing WWMCCS system interoperability. The criteria and techniques presented are indicative of the standards "discipline" involved in the evaluation and implementation of more effective standards for a complex computer network environment.

1.3 Scope

The evaluation and analysis herein focuses exclusively upon computer software required for the effective operation and use of a resource-sharing computer network. Hardware interface specifications and communications facilities per se were outside the scope of the project and, as the province of computer manufacturers and communications common carriers, are relatively beyond control of ADP system operators and users. The software functions examined, however, span the range of consideration in establishing an integrated computer network design, ranging from line control interactions in passing messages between computers to the interaction between a human user and software for networking ADP services. The standards issues raised by resource-sharing as suggested by the Prototype WWMCCS Intercomputer Network (PWIN) were the major consideration.

1.4 Approach and Organization of Report

The identification of key software interfaces for potential standardization is a guiding factor in the analyses performed, and is accomplished through a brief review of future WWMCCS network concepts in Chapter 2 following. As a result, the interfaces and interactions of interest separate naturally according to a categorization of functions, explained in Chapter 3, which distinguishes levels of control within a network. Chapter 4 comprises an evaluation of line control, the first level category which is most directly related to individual message transmission. Chapter 5 investigates the feasibility of common communications software for front-end processors which handle the second and third functional categories - flow control and message control. Chapter 6 analyzes the user-oriented protocols necessary for network ADP applications, involved in the last category. Chapter 7 reviews the techniques and conclusions from an overall standpoint, leading to a description of possible future work in standards analysis.

The reader is assumed to be familiar with WWMCCS and its current status, and with the design and operation of a network such as ARPANET. The background in WWMCCS underlying this report was provided in discussions with the sponsor representative, Mr. Kroboth, and in documents furnished by him.

2. WWMCCS COMPUTER NETWORK MODEL

The presently operational HIS* 6000 installations of WWMCCS rely upon a variety of communications arrangements for remote interactive and batch processing. To a large degree, these employ the HIS Datanet 355 as a front-end communications processor, with voice-grade lines to remote terminals. Communications interfaces to accommodate interconnection to the AUTODIN network of the DOD are being developed. Furthermore, the PWIN is being used as an experimental testbed to develop requirements for a future intercomputer network. Figure 2-1 suggests the variety of

*Honeywell Information Systems, Inc.

potential arrangements at an individual computer.

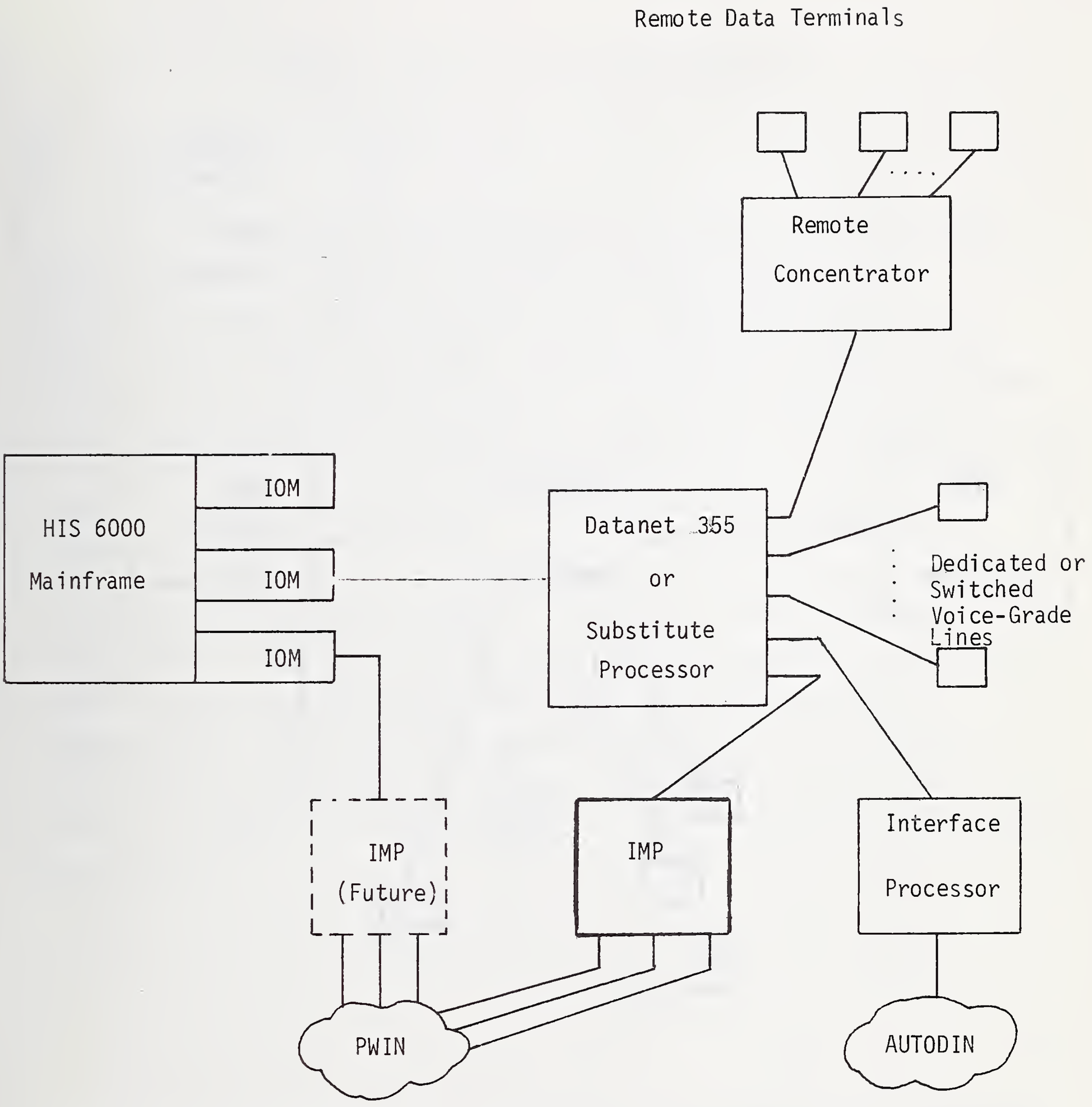
2.1 PWIN and ARPANET

The PWIN is closely modeled after the well-known ARPANET, utilizing three Interface Message Processors (IMPs) to interconnect three HIS 6000 sites for experimental operation. The PWIN project involves some significant design issues in the envisioned military uses of resource-sharing, e.g. the attainment of communications security and computer access control [2.1]. Also, because of its homogeneous computer hardware, PWIN has been able to adopt different techniques on a few special problems such as interfacing the IMP to the Host at the hardware level. Further, with a central planning group, PWIN has begun to define specific operational experiments and user-oriented software services to evaluate resource-sharing within the envisioned operational framework for WWMCCS.

Nevertheless, the strong similarities between PWIN and ARPANET, in conceptual, technological, and operational aspects, mean that inferences on standards drawn from consideration of ARPANET may very likely be applicable to the future WWMCCS.

2.2 Integrated Data Network

As a DOD system, WWMCCS will depend upon the common-use communications media provided as the Defense Communications System (DCS) by DCA. DCA is now planning toward a packet-switched data communications network, called The Integrated Data Network, IDN [2.2]. Connection to this network would be via a Communications Access Processor, providing many of the switching functions of the current ARPANET IMPs. Even so, there may be a continued need for a front-end programmable communications processor within WWMCCS - e.g. to handle locally connected data terminals or to implement higher-level standard protocols that are now performed within host computers or between hosts and IMPs. Thus, continued need for a WWMCCS - unique interface processor is a possibility, although not mandatory depending upon the IDN design. Therefore, figure 2-2 depicts the potential communications configuration envisioned in the future WWMCCS.



IOM = Input-Output Module of HIS 6000

Standard peripherals not shown.

FIGURE 2-1. Illustration of Potential Communication Arrangements at a WWMCCS Site.

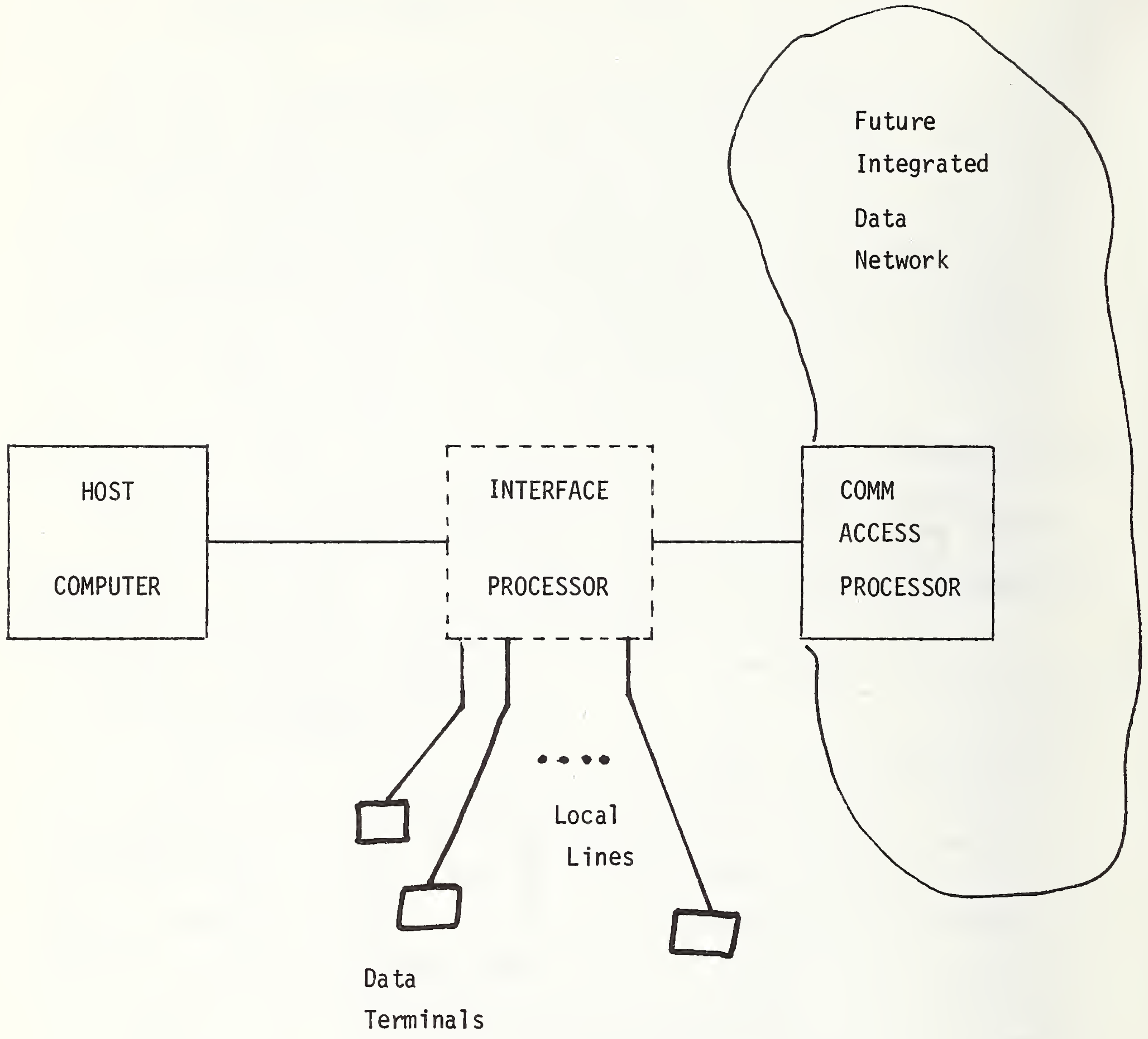


FIGURE 2-2. Future WWMCCS Network Model.

3. FUNCTIONAL ANALYSIS FOR NETWORK STANDARDS

3.1 Introduction

As data processing and data communications systems become increasingly integrated (as in the currently developing computer networks), the distinction between these two systems becomes increasingly difficult to make. Functions which were formerly thought of as purely communications functions, such as routing and message delivery, may be found being performed by essentially data processing components, while, conversely, functions formerly thought of as data processing functions such as scheduling and accounting may be found in the data communications system. This inherent flexibility of new computer communications systems to assign functions to different components gives increased importance to the analysis and implementation of effective standards. A first step toward standards development is a functional analysis which clearly identifies the functions which must be performed and maps them against software (or hardware) components where they are (or may desirably be) carried out. The result is an organized, unambiguous definition and a framework for examining standards trade-offs and potential impact on interfaces and interoperability.

3.1.1 A Model for Analysis

Figure 3-1 illustrates a model to be used for the functional analysis of a computer network system. The model recognizes that the essential function of the system is to transport a message as rapidly and reliably as possible from a sender to a recipient. To use the postal system as an analogy, a message may be thought of as a letter to be sent to a recipient. The content of the letter or semantics of the message have meaning only to the sender and the recipient; the intervening network components neither examine nor alter the message.

Between sender and recipient, the model identifies several stages of control or of function. There is a stage identified as "message control" which is concerned with the acceptance of the message from the sender by the network, and the delivery of the message from the network to the recipient. This is the stage at which an address is associated with the message. Continuing with the postal system analogy, at this stage the sender puts his letter in an envelope and addresses it.

The second functional stage identified by the model is called "flow control". This stage concerns itself with determining that the message is able to be delivered. Included at this stage are such network functions as call establishment, polling and dialing. In the analogy, the letter is put in a mailbox with sufficient room for it, and the pickup time and various internal delivery mechanisms such as train or plane are scheduled.

At the last stage there are the functions associated with the actual transmission media. This stage is identified as the line or device control stage. Actually, the procedures for controlling a computer network at this level are not so different from the procedures for controlling the postal delivery network. The messages (letters) are conveyed from point to point towards their destination, with care taken for their protection. At some final distribution point, packages of messages (letters) are turned over to a different level of control for individual delivery to each recipient.

3.1.2 Application of the Model

Applying this model to an actual system will require attention to the distinct operational modes of the system (which could coexist simultaneously) and to the fine distinctions between functions that are partially performed by separate components. After defining the generic function in more detail, two example applications will be made for WWMCCS. These illustrate the recognition of distinct modes (e.g. transaction processing versus remote batch) and distinct levels of interaction between components, which represent various levels of detail in use of this model.

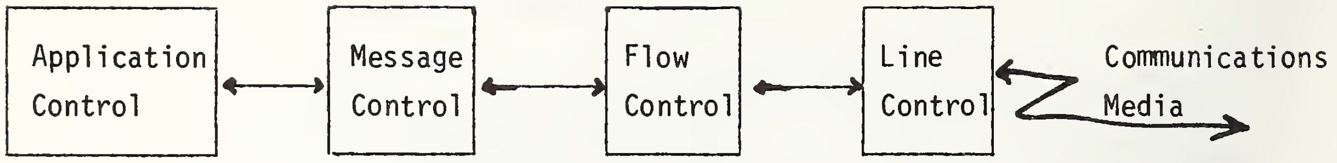


FIGURE 3-1. Functional Model of Network Software

3.2 Overview of Required Functions

3.2.1 Line Control

Line control is suggested as the generic term for those software functions concerned with the physical flow of data over the communications line or other media. The basic functions performed at this level are concerned with the editing of logical messages for physical output (and conversely or input), the initiation of input/output operations and error detection.

In his book, SYSTEMS ANALYSIS FOR DATA TRANSMISSION, [3.1], James

Martin lists twelve "basic functions", most of which would be included in the line control class:

1. To initiate and control the reception of data from the lines.

2. To assemble the bits into characters and the characters into messages.

3. To convert the coding of the characters (into the form used by the computer).

4. To check for errors, both in the characters, by means of a parity check, and in the messages, by means of longitudinal redundancy checks (where applicable).

5. To edit the messages if necessary.

6. To recognize end-of-record or end-of-transmission characters, and carry out the housekeeping, preparing for another transmission if necessary.

7. To deliver messages to the main programs, one at a time, edited and converted. (We do not consider this to be a line control function.)

8. To accept messages from the main programs when they are ready for transmission to the terminals. (We do not consider this to be a line control function.)

9. To prepare these messages for output. It will be necessary to convert them from computer code to communication line code. Control characters may have to be added.

10. To initiate the transmission of these messages.

11. To monitor the sending process, repeating characters or messages if the terminal (recipient) detects an error in transmission.

12. To signal end-of-transmission to the terminal and carry out the necessary housekeeping functions and line control functions.

As Martin recognizes, many of these functions may be performed either by hardware or by software. Functions most frequently implemented in hardware include bit assembly, error checking by horizontal and longitudinal parity, and the recognition and generation of such special characters as end-of-record and end-of-transmission.

3.2.2 Flow Control

In the category called "flow control" we include those functions concerned with call establishment and the scheduling and allocation of resources (such as buffers).

Martin lists certain polling and dialing functions which we include here since they may be considered as call establishment functions:

1. Dialing terminals.
2. Scanning dial up terminals.
3. Answering when the computer is dialed.
4. Polling by programming.
5. Polling with an autopoll device.

6. Controlling looped lines.

We recognize that many of these functions could be considered as line control functions since they are concerned with the detailed physical characteristics of devices. Since they are also directly concerned with call establishment, we arbitrarily assign them to the flow control class.

The heart of flow control is not call establishment, however, but resource allocation and scheduling. While the physical communications media may set upper bounds on the achievable data transfer rate, the actual rate attained is frequently limited by such matters as the amount of buffer space available and the priority level assigned to handling traffic. Normally, buffers are assigned at call establishment time so that they may be adjusted to the type of traffic expected and the current load level on the computer system.

In sophisticated systems where (parts of) messages may be received out ot sequence, elaborate protocol schemes may be employed to assure that buffer "lockup" conditions do not occur.

3.2.3 Message Control

Message control functions provide the software interface between application-oriented modules and the data communications-oriented modules. The primary function is to accept messages for delivery from a number of senders and pass them to the communications modules, and to deliver messages to the proper recipient which have been received through the communications system. Other functions such as spooling, packetizing and accounting may also be considered as message control functions.

3.2.4 Application Control

The application control level is the only level where message semantics are analyzed. All of the other levels were concerned with the delivery of the message. This level, in contrast, is concerned with the message content itself. Control may be exercised through common routines which implement protocols for such applications as file transfer or remote job entry, or it may be exercised strictly by the individual application programs to which the messages are delivered. Many logical levels of application control may frequently be nested in a single system.

3.3 Alternative Functional Breakdowns

The classes proposed in this document are not the only ones which could be used in the functional analysis of network software. Reference has already been made to James Martin's book where many communications functions are listed. Two other schema for classifying all network functions are provided in a book by Hal Becker, and a report by Planning Research Corporation.

3.3.1 Hal Becker

Hal Becker, in his book, FUNCTIONAL ANALYSIS OF INFORMATION NETWORKS[3.2], has suggested an alternative approach to the functional breakdown of network suggested here. Becker's approach provides a top-down taxonomy for all network functions which can be identified. No consideration is given to different logical levels of data flow. Becker's levels consist strictly of levels of implementation. Six levels are described, as follows:

I. Network - Any network function including both information and network processing.

II. Processing - Initial separation into distinct information processing and network processing functions.

III. Macro Function - Further separation of processing level into their respective hardware and software functions.

IV. Micro Function - Identification of basic hardware and software forms.

V. Element - Identification of specific hardware and software forms.

VI. Device/Technique - Identification of specific hardware devices and software techniques.

3.3.2 Planning Research Corporation (PRC)

A PRC report prepared for the Defense Communications Agency [3.3] suggested the categorization of network functions into the following classes:

1. Network/Line Control Management
2. Message Queue Management
3. Message Editing Management
4. Error Recovery Management

Similarities with the functional classification proposed in this document are evident in the PRC schema. Network/line control management includes approximately the same functions as device control. Message queue management includes many of the same functions as flow control. Message editing management includes some functions classified here as message control and some as application control. Error recovery management does not correspond to any specific classification in this document, since errors are presumed possible at any level and error handling is considered as part of the functional description in each class.

3.4 Functional Analysis for Different Network Organization

In this section we describe two separate networks in detail according to the functional schema which have been developed. The networks to be described are the typical remote processing networks supported by a Honeywell Information Systems Series 6000 system, and the ARPANET.

3.4.1 HIS 6000 Star Network

Networks are generally configured for a Series 6000 system by including one or more communications processors in the configuration. GECOS (the Series 6000 operating system) controls multiple communications processors (each with its own appropriate software), integrating remote processing with central-site processing in concurrent operation. Remote processing capabilities include remote batch, remote access, transaction processing, and time sharing [3.4].

3.4.1.1 Communications Processors

The three communciations processors that provide the front-end processing of Series 6000 Information Systems are:

- DATANET 355 Communications Processors
- DATANET 305 Communications Processors
- DATANET 30 Communications Processors

Only the DATANET 355 and the DATANET 30 are specified under the terms of the WWMCCS procurement.

3.4.1.1.1 DATANET 355

The DATANET 355 is a high-performance, stored program communications processor designed to service large volume communications needs of the Series 6000 System. A DATANET 355 may have a memory size of 16K or 32K 18-bit words with a cycle time of one microsecond. Data of various word lengths may be processed -- 6, 9, 18 or 36 bits. All data word lengths are individually addressed to allow efficient processing of tabular data. Ninety-eight instructions in an 18-bit format are provided, with one single-address instruction per word. Three index registers and multilevel indirect addressing, with indexing at all levels, give an addressable storage capacity of 32K words. A sophisticated priority interrupt system is provided.

The system organization of the DATANET 355 follows the pattern of the Series 6000 system architecture. The DATANET 355 is a memory-oriented computer with its own independent memory, processor, and I/O modules, all of which operate asynchronously. The processor and I/O controller are "active" units that process data at their own rates and request memory cycles as the need arises. Input/output facilities of the DATANET 355 are designed to facilitate efficient real-time, concurrent servicing of multiple terminals and peripheral devices. Up to 16 adaptors can be provided to accommodate a total data rate of up to 500,000 words per second (with 6, 9, 18 or 36 bits per word). The following adaptors can be configured:

1. DATANET 355 Intercomputer Adaptor (ICA) with up to four ICA ports to interface with the Series 6000 system controllers.
2. Up to three High-Speed Line Adaptor units (HSLA).
3. Up to six Low-Speed Line Adaptor units (LSLA).
4. A console adaptor for connection of a teletypewriter console.
5. An adaptor link to Series 6000 mass storage.

When attached to a Series 6000 system, the DATANET 355 operates under a complex of software routines collectively referred to as the General Remote Terminal System (GRTS). GRTS consists of two main parts, GRTS-355 which executes in the DATANET 355 and GRTS-6000 which executes in the Series 6000 mainframe. GRTS-355 is naturally designed to communicate with GRTS-6000, and GRTS-6000 is designed to interface with GCOS, the General Comprehensive Operating System of the Series 6000.

3.4.1.1.2 DATANET 30

The DATANET 30 is the other communications processor that is part of the WWMCCS configurations. The DATANET 30, like the DATANET 355, is a stored program communications processor that provides front-end communications processing for the Series 6000.

The DATANET 30 has 16K words of memory with a 7 microsecond cycle time. The processor is designed to handle 18-bit words and allows up to 60 I/O channels. Where the DATANET 355 is connected directly to the system controller, the DATANET 30 is connected to the IOM and IOC as any other common peripheral device.

Two software packages are available for the DATANET 30: a mixed speed package and a low-speed package. The mixed-speed package will service any one of the following at any time:

1. 31 teletypewriters (110 to 150 BPS).
2. 10 remote batch computers (voice grade speeds).
3. 4 remote batch computers (wideband).
4. 4 CRT terminals (voice-grade).
5. A combination of the above devices.

The low-speed program will service up to 60 teletypewriters (110 to 150 BPS).

3.4.1.2 Remote Batch Mode

Any job that can be entered directly at the central system can be entered remotely from a remote batch computer. A remote batch job differs from a local batch job only in the GECOS job input/output routines that interface with the communications processor. Once inside the central system, local batch and remote batch processing are identical.

3.4.1.2.1 Line Control (Remote Batch)

Remote batch jobs are expected to be submitted from a Honeywell 115 remote batch station. The H-115 communicates with the Series 6000 communications processor according to a binary synchronous line discipline. Any other remote batch station can submit jobs so long as it can conform to this line discipline. At the Series 6000 side, all line control functions are provided by the communications processor, through the combination of its software and communications adaptor.

3.4.1.2.2 Flow Control (Remote Batch)

An incoming remote batch message stream from a Honeywell 115 or compatible terminal is buffered in the DATANET with identifying control information. As the buffer fills, it is transferred by GRTS-355 to GRTS-6000, which places it on mass storage and informs GCOS (when the complete job has been stored) so that the job may be scheduled for execution.

3.4.1.2.3 Message Control (Remote Batch)

Remote batch jobs have several options available for the output files they generate. These options are selected by control cards as follows:

Return to sending terminal
Enter into the file system.
Output at central site.
Hold until terminal calls back.
Send to another terminal.

3.4.1.2.4 Application Control (Remote Batch)

Incoming messages represent either jobs to be compiled and run or data for such jobs; outgoing messages represent the printout from running these jobs. Jobs submitted are queued for execution in the normal job stream; printouts returned to the originating terminal are printed there.

3.4.1.3 Remote Access Mode

Remote access provides direct terminal access to a program in execution. The program, written in any batch programming language, can be submitted as a job via local batch, remote batch, or time-sharing. The terminal in remote access effectively becomes an on-line peripheral to the activity in process. The activity can send output to and receive input from the terminal or terminals.

3.4.1.3.1 Line Control (Remote Access)

Terminals such as teletypewriters and compatible alphanumeric displays operate asynchronously and generally at relatively low speeds (30 characters per second and under). Line control functions for such terminals are implemented principally in hardware in the high-speed line adaptor (HSLA) or low-speed line adaptor (LSLA) on the DATANET 355, and similarly in the DATANET 30. The LSLA provides the primary facility for connecting low speed terminals to the DATANET 355. The LSLA operates with low speed terminals in either full or half duplex mode for asynchronous data transfer. The LSLA operates on the principle of time-division multiplexing, developing a message frame composed of a number of 8-bit characters, called time-slots, where each time-slot contains one complete character associated with a particular terminal.

3.4.1.3.2 Flow Control (Remote Access)

Flow control in remote access mode is provided by the application programs to which the terminals are logically connected.

3.4.1.3.3 Message Control (Remote Access)

Once a terminal has been placed in remote access mode with a user program, messages from the terminal are sent directly to the program by GRTS.

3.4.1.3.4 Application Control (Remote Access)

Application control for remote access processing is provided entirely by the program to which the terminal is connected.

3.4.1.4 Transaction Processing Mode

A Transaction Processing Executive controls the concurrent execution of application programs under GECOS, providing on-line real-time capability. The executive operates as a privileged slave in the direct access mode and will generally be in execution. The application programs will not generally be in execution, but will be activated by the executive selectively upon receipt of transactions (remote user inputs requiring immediate servicing). The transaction processing application programs are normal user application programs and may be written in any of the Series 6000 languages.

3.4.1.4.1 Line Control (Transaction Processing)

Line control functions for transaction processing terminals are provided as described in section 3.4.1.3.1 for the line control of remote access terminals. Some additional functions concerning the compacting of messages are also performed for transaction processing. These are described in context below in section 3.4.1.4.2.

3.4.1.4.2 Flow Control (Transaction Processing)

Flow control is provided by the Transaction Processing Executive (TPE). Messages arrive first at the DATANET 355, where the characters are collected in a buffer. GRTS adds the control characters that inform the H6000 of message origin, unpacks messages that may have been compacted at the terminal, and sends message segments of 324 characters (maximum) to the H6000.

If the terminal is not in direct access mode with the application program, the TPE must first write a journal record of the message, do any necessary code conversion, and then spawn the appropriate application program. (Spawning an application program would be considered message control.) If a higher priority message is ahead of this message, the message is put in a queue for later processing. Low priority messages are processed on a "time available" basis.

If the terminal is in direct access mode, the TPE is bypassed after the first message is received from the terminal. This means that the communications executive functions must be performed by the application program. In all cases, however, checkpoint writing must be done by the application program, since the TPE does not do it.

When the application program sends a response to the terminal, the logic is reversed from the input. If in direct access mode, the application program must write a journal record and checkpoint, and then segment the message to send it to the DATANET 355. If not in direct mode, the TPE writes the journal and segments the message. In the DATANET, GRTS compacts the message to remove redundant characters and places it in the appropriate output buffer. If the line is free, the message then is sent to the remote terminal. (These last functions are line control functions.)

3.4.1.4.3 Message Control (Transaction Processing)

As described in section 3.4.1.4.2, message control for transaction processing is provided either by the TPE or by GCOS. The TPE provides message control for the initial correction by spawning a job, if necessary. After that, or if the terminal and program were already in direct mode, GCOS sees to it that terminal messages are delivered to the proper application program.

3.4.1.4.4 Application Control (Transaction Processing)

Application control for transaction processing is provided entirely by the application programs to which the terminals are connected.

3.4.1.5 Time Sharing Mode

A Time Sharing Executive performs the functions of selecting, allocating, dispatching, and swapping time-sharing user programs. The executive is structured as a single privileged slave program operating under the control of GECOS. It, in turn, suballocates memory and subdispatches the processor to individual time-sharing user programs. The Time Sharing Executive also performs various services for individual programs, including file system I/O, terminal I/O, and creation and modification of files and catalogs.

3.4.1.5.1 Line Control (Time Sharing)

Line control functions for time sharing terminals are provided as described in section 3.4.1.3.1 for the line control of remote access terminals.

3.4.1.5.2 Flow Control (Time Sharing)

There is essentially no flow control for time sharing terminals, since they are asynchronous and may transmit a character at any time. Clusters of terminals are configured on the various line adaptors according to the speed of the terminals, so that characters will not be lost due to overloading at that point. However,if the software cannot keep up with the inflow of characters from the terminals, some may be lost.

3.4.1.5.3 Message Control (Time Sharing)

The Time Sharing Executive is responsible for delivery of messages to the proper user program, and for indicating to GRTS the originating user program of messages so that they may be sent to the proper terminal.

3.4.1.5.4 Application Control (Time Sharing)

Application control for time sharing is provided by each of the processors and/or programs to which the terminals are connected.

3.4.2 ARPANET

The ARPANET is a large packet-switching network of distributed host computers, switching processors and high speed communications lines which was sponsored by the Department of Defense Advanced Research Projects Agency. The switching nodes, called Interface Message Processors (IMPs) are connected to hosts for the acceptance or delivery of messages from or to other hosts, and they are connected to each other for the forwarding of messages to their ultimate destination [3.5]. Messages which are passed to the IMPs from the hosts for transmission through the network are broken up by the IMPs into smaller transmission blocks called packets. Each of the packets of a message is routed individually (and perhaps differently) towards its destination, where it is reassembled by the final IMP and delivered to the proper host. Software in the host referred to as the Network Control Program (NCP) accepts the message from the IMP and delivers it to the proper user process [3.6].

The ARPANET is a very complex communications network with different types of physical and logical transmission occurring between different points. The levels of transmission are generally described in terms of the various levels of protocol which have been implemented. We will apply the functional analysis schema here to each of these levels of protocol.

3.4.2.1 Third-Level Protocol (Process to Process)

In the ARPANET, certain application control functions are standardized as so-called third level protocols [3.7]. No one is constrained to use these protocols (private protocols are permitted and even encouraged for experimentation), but many of them have achieved wide acceptance. The TELNET protocol, for example, which provides for a common format for representing messages from alphanumeric communications terminals, is subscribed to by nearly all sites. The initial connection protocol is also very widely used. Other protocols of this type include (at varying levels of complete definition and acceptance) data transfer protocol, file transfer protocol, remote job submission protocol and network graphics protocol.

These protocols all must concern themselves with message semantics. Even the initial connection protocol, which might appear to be a communications-oriented function, only utilizes lower level functions (from the ARPA second level protocol) to accomplish its tasks. Of course, if the communications model is to be applied to the virtual communications between hosts (and all intervening elements are to be considered as a black box), then we could indeed consider the initial connection protocol to be performing the flow control function.

3.4.2.1.1 Line Control (Process to Process)

Line control mechanisms at the process to process level are not realized in hardware, but in the software procedures by which processes access the operating system and/or the network control program. (This access mechanism could involve the use of special instructions such as "executive request" or "supervisor call"). These mechanisms are likely to differ from host to host. The function performed, that of entering messages into the communication system and initiating their transmission, is entirely analogous to the functions performed by low level software control of a hardware port or channel.

3.4.2.1.2 Flow Control (Process to Process)

There is no explicit flow control mechanism at the process to process level. Individual host operating systems may limit the rate at which processes may enter messages into the network and may refuse to accept messages at particular times, but these are local conventions only. Similarly, particular third-level protocols may establish conventions regarding flow control for that application, but these conventions do not hold for third-level protocols in general.

Note that the flow control which is imposed by the subnet for message traffic between hosts is considered in section 3.4.2.2.2 and not here, since it is not a flow control mechanism for messages between processes, but between hosts.

3.4.2.1.3 Message Control (Process to Process)

Message control functions are provided by routines in the host's operating system or network control program. The operating system must provide entry points for individual processes to send and receive messages. Since operating system routines are normally reentrant, provision must be made for logically multiplexing messages between the operating system and the network. All this really means is that the operating system must maintain tables to permit it to associate network addresses with local processes so that messages may be delivered to their intended recipient.

Communications between processes and the operating system will be in terms of messages of up to 8095 bits in length, including control information such as message address. In addition, there may be local conventions for operating system linkages, transfer of control, and error recovery routines. These conventions are likely to be different for each host system.

3.4.2.1.4 Application Control (Process to Process)

At this level of protocol, message semantics are analyzed for the purpose of application control. Each different protocol at this level will have its own formats and codes for communicating information and control. Communications between processes at this level are no different than if the processes were located in the same host computer and communicated through the operating system.

3.4.2.2 Second-Level Protocol (Host to Host)

The second-level or host to host protocol is concerned with the exchange of messages (as opposed to simply packets) between the operating systems or executive level routines in each host. Each operating system treats the entire subnet as a "black box" communications device accessible through the host-IMP interface. The operating system multiplexes messages which may originate from various processes in the host and dispatches them through that interface. The communications process within the subnet is of no concern at this level.

3.4.2.2.1 Line Control (Host to Host)

Hosts communicate among themselves in terms of variable length messages which may be up to a maximum of 8095 bits in length (including control fields). The second-level or host to IMP protocol is considered as the line control procedure at this level.

3.4.2.2.2 Flow Control (Host to Host)

The semantics of the second-level protocol is largely oriented towards providing flow control for third-level messages. There are commands for opening and closing logical connections, allocating and releasing buffers for communications according to specified message sizes, and for various test and error recovery capabilities. In communications jargon, these commands perform the call establishment function for process to process communication.

Flow control for the second-level messages themselves is provided by the conventions for host to host protocol between network control programs in different hosts, and by subnet flow control which applies to all messages.

The semantics of the second-level protocol indicates which commands must be used in answer to which other commands. For example, a request for buffer allocation must be answered by acceptance or refusal of the request. Thus, the requirements to perform transactions in a specified order provide a sort of flow control at the host to host level.

More forceful flow control over physical messages is imposed by the subnet. Only four messages are permitted to be in transit at any one time between any pair of source and destination IMPs. (Recall that up to four hosts can share any IMP). This flow control mechanism is imposed without regard to the level of information conveyed in the message.

3.4.2.2.3 Message Control (Host to Host)

Message control functions are provided at this level by internal linkages in the network control program between the routines which realize the host to host protocol and routines which control the host-IMP interface. Messages are delivered to the proper host by the subnet according to addresses passed from the second-level to the first-level protocol routines and inserted in the proper field in the message leader.

3.4.2.2.4 Application Control (Host to Host)

Application control is accomplished by the use of the commands provided by the host-host protocol:

Connection Establishment - Two commands are provided for the purpose of establishing a one-way logical connection. These commands identify sending and receiving internal addresses and connection byte size. The commands must be exchanged between sender and recipient.

Connection Termination - A single command is provided for closing connections. Each side must send and receive this command before the connection is terminated.

Flow Control - Three commands are provided for allocating buffers for the purpose of flow control. One command requests a buffer allocation, a second command grants it, and a final command releases the buffer.

Interrupts - Two commands are provided for sending an "interrupt" by either sender or receiver. The meaning of the interrupt is not defined at this level, but is available for use by higher levels of protocol.

Test Inquiry - A command and a reply are provided for reinitializating connections. The semantics of this command pair may also provide direct flow control at the host to host level by suspending communications over a correction period of time.

Errors - A command is provided for conveying error information to the hosts.

3.4.2.3 First-Level Protocol (Host to IMP)

Most hosts in the ARPANET are connected to local IMPs by means of a "special host interface" which is a high speed serial interface designed to permit dissimilar hosts to be connected to IMPs in a relatively standard manner. Data are exchanged through this interface a bit at a time at a rate determined by the host in question.

A "very distant host" interface is also supported which permits a host to interface with an IMP through a communications line in essentially the same way that IMPs are connected with each other. In this case, the host and IMP communicate using a binary synchronous line discipline and at a rate determined by the modems employed.

Messages between host and IMP can be for the purpose of dispatching or receiving messages to or from other hosts or to exchange control information between host and IMP.

3.4.2.3.1 Line Control (Host to IMP)

The special host interface is implemented as a standard portion which is built into each IMP and a non-standard portion which must be designed for each different type of host. The interface is designed to allow messages to flow in both directions at once. A bit serial interface was designed partly because it required fewer lines for electrical interfacing and partly to accommodate the different word lengths of different host computers.

The host interface operates asynchronously, each data bit being passed across the interface via a "Ready for Next Bit/There's Your Bit" handshake procedure. This technique permits the bit rate to adjust tothe slower member of the pair. A special utility routine or access method must be coded for the host to control this interface. Hosts may have this control routine implemented in different ways, so no generalizations may be made about the line control strategies at this level in the host.

When a very distant host interface is employed, line control functions are accomplished in the same manner as for IMP-IMP communication. This is described in section 3.4.2.4.1.

3.4.2.3.2 Flow Control (Host to IMP)

Due to the asynchronous nature of the interface between the host and the IMP, there are no flow control problems at this level. If either host or IMP is unable to accommodate incoming messages at a particular time, either may inhibit the flow of data through the interface by itself.

In the case of a very distant host interface, flow control is accomplished in a manner similar to that employed for IMP-IMP communication (described in section 3.4.2.4.2). However, for the very distant host interface, there are only two logical channels established, so that only two unacknowledged messages may be outstanding at any point in time.

3.4.2.3.3 Message Control (Host to IMP)

Message control requirements at the host-IMP level are minimal. The IMP must have the ability to recognize messages from different hosts, if there is more than one host connected to it, but each host communicates only with its own IMP at this level.

3.4.2.3.4 Application Control (Host to IMP)

Application control functions at the host-IMP level are indicated by a 32-bit leader which occupies the first part of the text of all messages between host and IMP. The leader contains fields which indicate the type of the message and its destination. (Messages may be destined for some distant host or for a fake host in the IMP.) Message types may be regular messages, requests for next message (RFNM), or various types of error messages. Programs in the host and in the IMP (routines in the network control program in the host and routines in IMPSYS in the IMP) analyze the content of the leader and take appropriate action.

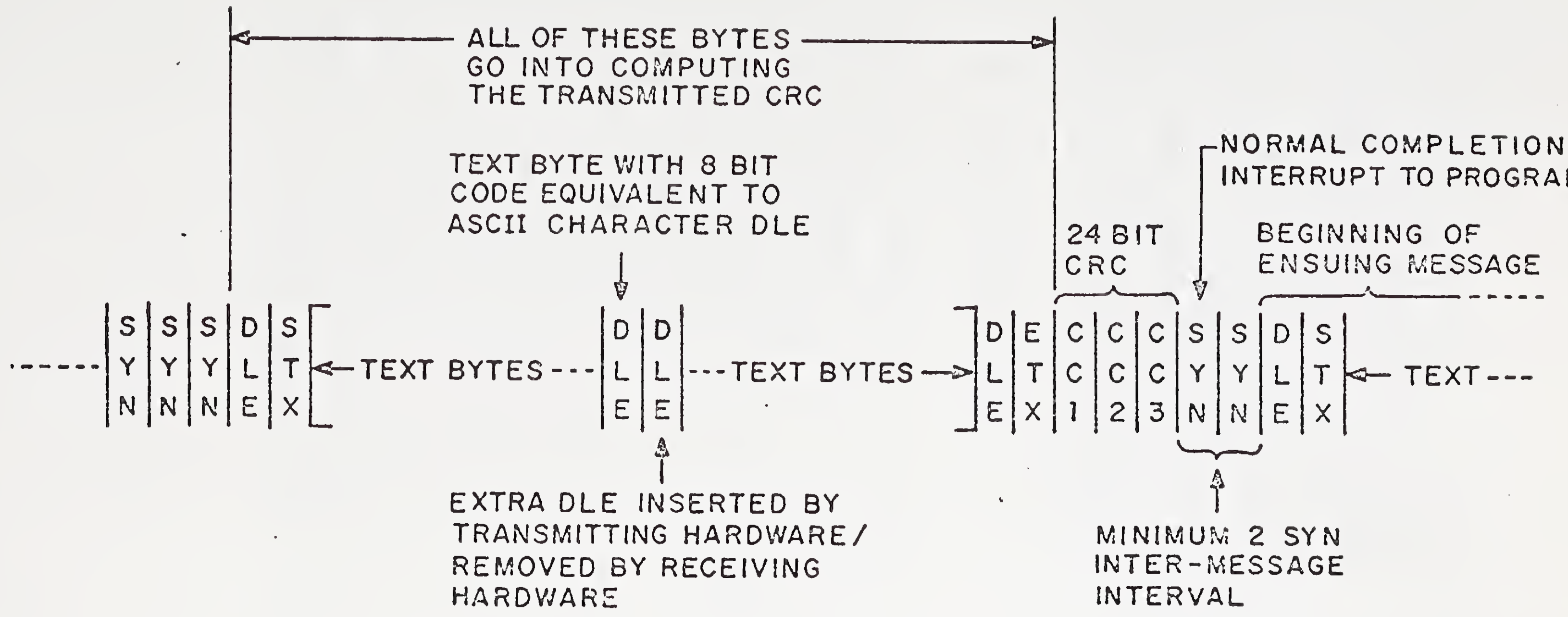
3.4.2.4 Zero-Level Protocol (IMP to IMP)

The IMPs in the subnet communicate among themselves for the purposes of forwarding messages originating from the hosts and exchanging routing and other control information. Most IMPs are connected to at least two other IMPs by means of terrestrial leased lines operating at 50 Kbps; however, some IMPs are connected by lines operating at other speeds, both faster and slower, and some satellite links are in use. Modems are in use at the end of nearly all lines and a binary synchronous line control procedure is used for the IMP to IMP data exchange.

3.4.2.4.1 Line Control (IMP to IMP)

The IMPs in the subnet forward messages from the hosts and exchange routing and control information in units called packets. With the exception noted below, packets are limited in length to 1024 bits of text (including control fields). Packets are transmitted between IMPs as the text portion of a physical transmission block conforming to the format shown in Figure 3-2.

Special hardware is included in the line interface units of the IMPs to facilitate the use of this line control procedure. Specifically, the text framing characters (DLE STX and DLE ETX) are recognized by hardware, embedded DLE characters are handled by hardware, and the cyclic redundancy checksum is computed by hardware. The doubling of DLE characters which occurs as part of the test of the packet (in order to distinguish them from control characters) provides the exception to the 1024-bit packet length limitation. Packets may be up to nearly twice as long if many DLE characters occur naturally as part of the packet text; however, this extra length will only be evident during


ALL OF THESE BYTES GO INTO COMPUTING THE TRANSMITTED CRC
TEXT BYTE WITH 8 BIT CODE EQUIVALENT TO ASCII CHARACTER DLE
24 BIT CRC
NORMAL COMPLETION INTERRUPT TO PROGRAM
BEGINNING OF ENSUING MESSAGE
SYN SYN SYN DLE STX
TEXT BYTES
DLE DLE
TEXT BYTES
DLE ETX CC1 CC2 CC3 SYN SYN DLE STX
TEXT
EXTRA DLE INSERTED BY TRANSMITTING HARDWARE/ REMOVED BY RECEIVING HARDWARE
MINIMUM 2 SYN INTER-MESSAGE INTERVAL

PACKET FORMAT ON LINE

FIGURE 3-2

transmission and not in the memory of either the transmitting or receiving IMP, since the extra DLE characters are inserted and removed by hardware as part of the transmission process.

Control of the lines and their interfaces is effected by the program in each IMP which designates input and output buffers for each line/interface. The program is highly interrupt driven; once an input/output operation has been initiated on a particular line, it continues without active processor intervention until completion, when the processor is interrupted and notified.

3.4.2.4.2 Flow Control (IMP to IMP)

Two flow control mechanisms are relevant at the IMP to IMP level. There is a mechanism for governing the flow of any type of packets between adjacent IMPs, and a separate mechanism for governing the flow of host message packets between source and destination IMPs (the IMPs connected to the source and destination hosts, respectively).

The line control procedure between IMPs requires that all packets be acknowledged for each hop (transmission from IMP to IMP). However, for purposes of maintaining a high rate of data flow to make efficient utilization of the full duplex lines, the flow control system tries to avoid waiting for acknowledgments. The mechanism employed permits up to eight unacknowledgments messages to be outstanding at any given time. By reserving a bit in the packet header a logical control field at the start of each packet's text (field) for each possible outstanding message, several messages may be acknowledged at once. Packets are automatically retransmitted when it is noted that they have not been acknowledged.

The order of packets is only of significance to sending and receiving IMPs, since a message, which may consist of up to eight packets, must be reassembled at the receiving IMP before delivery to its host. A special flow control mechanism was instituted between sending and receiving IMPs when it was realized that insufficient buffer space in the receiving IMP to reassemble all the packets of a single message (because that space was occupied by the packets of other partially assembled messages) could crash the network. The mechanism requires that reassembly space for multi-packet messages be reserved in advance, before all the constituent packets are released into the subnet by the sending IMP. For the sake of efficiency, the request for a buffer allocation is sent "piggyback" with the first packet of the message. For simplicity in buffer management, space is allocated in eight packet blocks if it is available, and notice of such allocation is sent back to the source IMP, at which time the remaining packets are released. If no space is presently available, no allocation notice is sent, and the source IMP must try again.

3.4.2.4.2 Message Control (IMP to IMP)

Each packet transmitted between IMPs in the subnet is a complete message so far as the IMPs are concerned. That message either tells the IMP that it must forward data to another IMP, or it contains control information for one of a number of special programs in the IMP. A field at the start of the text portion in each packet identifies the packet as one to be forwarded or for which IMP program it is intended. This identification of packet type provides all the message control which is needed at this level.

3.4.2.4.3 Application Control (IMP to IMP)

Packets transmitted from IMP to IMP are either to be passed on until they reach the destination IMP for reassembly and delivery to the host, or they are intended for one of a number of special background programs in the IMP itself. The background programs are each treated as a "fake host" so that selected hosts and IMPs can communicate with them and so that existing routines in the IMP can be used to process this type of message. A "For IMP" bit or "From IMP" bit in the leader (the control field at the start of the text of each packet) indicates that a particular message is to or from a fake host program in the IMP.

The background programs perform a variety of functions: TTY is used to handle the IMP Teletype traffic; DEBUG, to inspect or change IMP core memory; TRACE, to transmit collected information about traced packets; STATISTICS, to take and transmit network and IMP statistics; and DISCARD, to throw away packets. Other routines, which send incomplete transmission messages, send allocations, return givebacks, and send RFNMs, also reside in the background program.

Bits in the control field at the start of the packet text (the leader) indicate which background program is associated with the message. The text of the message is naturally application dependent.

3.5 Interface Considerations for WWMCCS

The foregoing illustrates that the suggested model is useful in developing comparisons of functions accomplished in the different modes and components in the present WWMCCS. A complete and perhaps more detailed development would allow a continuing commonality analysis during the near-term evolution of WWMCCS (e.g. the replacement of DN 355 - GRTS by the projected NPS software, or alteration of the HIS 6000 - PWIN interface). Such analysis would permit identifying trends toward replication or incompatibility in functions at each system interface.

Returning, however, to the envisioned future WWMCCS network, figure 2-2, it is clear that comparisons should focus on the evolution toward the major interfaces: user to host, host to interface processor, interface processor to communications network. In the present WWMCCS, these interfaces exist in several forms[3.7,3.8]. For example, the host may interface to remote users through DN 355 - GRTS, through PWIN, or (soon) through AUTODIN. An economical and organized evolution toward the model of figure 2-2 would be guided then by analysis of the following potential standards:

(1) A standard line control discipline applicable to PWIN, future IDN, and possibly other switched, synchronous communications.

(2) A standard interface processor software concept, allowing flexibility in interfacing to several communications facilities as well as host software.

(3) Standard user-oriented software functions (protocols) for readily accomplishing resource-sharing objectives, despite the explicit communications implementation.

These standards seem promising in solving the interim interoperability problems, and meeting the ultimate configuration goal. Hence, they have been investigated further with results given in Sections 4.5, and 6.

4.0 ANALYSIS OF COMMUNICATIONS CONTROL PROCEDURES

4.1 Purpose and Scope

The transmission of intelligible information between a generator of and responder to that information is at present based upon a data communications control procedure* [4.1] that specifies appropriate transmission times and defines appropriate message structures. The character oriented discipline described in the ANSI Standard X3.28 [4.2] and based on ASCII Code, represents the culmination of efforts to use a subset of a character code for line control purposes. The proliferation of time-sharing terminals and the rapid development of computer networking has multiplied the complexity of the control problem to such an extent as to foster the development of a suggested bit-oriented control procedure. At the international level, this is embodied in the High-Level Data Link Control Procedures (HDLC) [4.3] and at the national level, in the very similar Advanced Data Communication Control Procedure (ADCCP) [4.4]. Some of the advantages of this bit-oriented control procedure are that it reduces the total number of bits needed for control, allows for a uniform format for and a broader variety of control information, lends itself to modularity in the addressing capability, and provides for explicit addressing of all commands and responses - a useful feature for achieving reliability in switched networks.

In recent years, the technique of Finite State Representation ([4.1, 4.5]), developed in the field of finite state automata, has been adopted for use in the description of these data communications control procedures. This tool enables one to specify, to as detailed a level as desired, the allowable control command/response sequences for two communicating stations. This specification can be made in the form of a state-event matrix or its equivalent graph. A particular station's communications software, which is executed between the occurrence of events (the sending of commands or responses), can be described and located in the appropriate

*Alternative names for this are: data link control procedure, line protocol, line discipline, or message discipline. The name used in this report follows ANSI terminology.

boxes of the state-event matrix, thus completely representing the logic accompanying the transmission between two stations. It shall be shown in the subsequent sections how this tool might be used to compare the complexity of the two disciplines (X3.28 and ADCCP) as used in the control procedures of a single data link in a packet communications network such as ARPA and PWIN.

4.2 Finite State Representation and Current Use of X3.28-1971 in ARPA and PWIN

The Finite State Representation technique describes a system in terms of a finite number of states* it can assume and a finite number of events (its alphabet of commands and responses) that can trigger its transition from one state to another. The discussion will now proceed to describe the systems and alphabets used in data communications with special emphasis on ARPANET and PWIN so that the method becomes clear.

4.2.1 The Data Communications Link Model: In data communications, the specific system under consideration is derived from some category of link configuration (multipoint or point-to-point, switched or non-switched) and directionality (one-way, two-way alternate [half-duplex], or two-way simultaneous [full-duplex]) among a set of communicating stations. These stations can be initiators of messages (a primary), responders to messages (a secondary) or both. When a station is either a primary or a secondary but not both, one is dealing with a centralized operation. When each station can be either a primary or a secondary, one has a decentralized operation.

At the most fundamental physical link level, all of the categories of communications systems can be described by means of a model that uses a primary link control (PLC), located at the primary, communicating with a secondary link control (SLC), located at the secondary. A centralized

*A state of a transmission system is defined in terms of its being ready to send or ready to receive a command or response.

primary/secondary communication with two-way alternate transmission can be represented by a model with two sublinks, as shown in figure 4-1a [4.4]. The information traverses the links alternately within the closed path shown. A decentralized primary/primary communication system also has two sublinks, but they are modelled as shown in figure 4-1b [4.4]. For two-way alternate, two closed paths are possible as shown. The decision as to which path is used in any transmission depends on which station assumes the role of the primary and which one becomes the secondary. Once this decision is made, the model is exactly the same as in figure 4-1a. For two-way simultaneous transmission, these two paths remain the same but must be interleaved along the sublinks properly.

It is the model of figure 4-1b that is appropriate for the IMP-IMP links in ARPANET and PWIN since transmission is always full-duplex and IMPs are always both originators of and responders to messages.

4.2.2 Finite State Graph/Matrix-Level 1: The alphabet of commands and responses chosen for a particular finite state graph or matrix is dependent upon the level of explicitness desired. For the simplest level (level 1 in reference [4-1]) of a system such as the two-way alternate link shown in figure 4-1a, one transmits an acceptable sequence of characters μ which causes a transition from state P (primary ready to send [generate] a message) to state S (secondary ready to send [respond to] message). When the secondary responds in the form of an acceptable string of characters υ, the system reverts to state P. Figure 4-2a shows what has occurred in the system in the form of a finite state graph while figure 4-2b contains its equivalent matrix. Any unexpected combination of alphabet and state is termed an error and denoted by E.

It should be noted that there is a fundamental duality imbedded in this model of the control procedure. When the primary is sending a message (a command), the secondary is receiving that message (the same command) and, similarly when the secondary is sending a message (a response), the primary is receiving that message (the same response). One can describe the states of the system either from the point of view of 'ready to send' or 'ready to receive'. However, this duality disappears when the communications software is introduced into the picture since the actions taken at the sender (by its software) must, in general, differ from those taken at the receiver (by its software).

The simplest level of the graph and matrix for the primary/primary link model in the two-way alternate case is that shown in figures 4-3a and 4-3b. Close examination shows that this case is just a linear superposition of two primary/secondary links finite state graphs. The case of two-way simultaneous primary/primary transmission requires the proper interleaving of the information traversing sublinks 1 and 2. This may be accomplished by the rule that when station 1 sends a message, station 2 sends either nothing, synchronization symbols, or a message of the same type. It is not necessary for the stations to maintain matching mutual character-by-character sequences. The finite state graph of the two interleaved half-duplex transmissions uses concepts found in parallel computation and Petrie nets. Since the differences between the X3.28-1971 and ADCCP protocols show up before reaching this level of complexity, implications of interleaving will not be discussed for the IMP-IMP protocols.

4.2.3 Finite State Graph/Matrix-Level 2: In the case of IMP-IMP transmission, the string of characters μ during message transfer phase is always a packet* of information and the string of characters υ is an 'ack' or acknowledgement of the packet. At the second level of explicitness, one can explode the P and S states into substates in which the system is ready to send different types of messages. For the ARPA and PWIN networks these messages can be one of three types of packets: store and forward packets (sfp), routing packets (rp), and null packets (np). The store and forward packets contain information that is being forwarded through the network; routing packets contain path length and delay time information for adjacent IMPS; and null packets contain acknowledgement

*The message at the Host-Host level of communication is always broken down into standarized packets (essentially a form of blocking) at IMP-IMP level.

information between communicating IMPS [4.7]. Store and forward packets as well as routing packets, additionally piggy-back acknowledgement information between IMPS to speed the traffic through the network. Store and forward packets and routing packets are sent by a primary, while null packets are sent by a secondary. See figure 4-4a and 4-4b for the second level graph of the primary/primary two-way alternate case. Note that there is no provision for acknowledging a routing packet or a null packet. Also, note that only a limited set of software actions has been specified in the matrix.

4.2.4 Finite State Graph/Matrix-Level 3: The third level of explicitness can be shown by exploding the sending of a packet (any of the three) into its component parts (see figure 4-5). The differences in the three packets can be found in the number of header words used. The protocol, on the line, used by ARPA and PWIN, contains four of the ten ASCII communication control characters (SYN, DLE, STX, and ETX). Interpacket synchronization is achieved with a minimum of two SYN characters; framing of a packet is achieved with the two character sequences DLE STX and DLE ETX. The first header word in a store and forward packet contains the logical channel number for that packet and any acknowledgements for these logical channels. There are currently eight logical channels from one IMP to the next. This logical channel number performs a link address function which allows the IMPs to tag the acknowledgements properly. The routing and null packets do not use a channel number since there is no acknowledgement to these two type packets. Although the earlier descriptions of the line protocol indicate that the 24 check characters (CRC) consist of a cyclic redundancy check, a more recent report by BBN [4.8] indicates that a faster, but less accurate checksum technique is actually in use. A protocol feature that has been added to produce transparency is the addition of DLE everytime a DLE is found in the packet (states 4 to 5 in figure 4-4a). The contents of the header words for the different packets can be found in reference [4.9].

Error control is implicit in this protocol in that bad packets and packets for which there is no buffer space are discarded with no acknowledgement. Before every retransmission, the checksum is recalculated to see whether the problem is an intra-IMP failure.

One could continue this process of expanding on the states of this system down to the bit level, but it is not necessary for comparing X3.28 (as implemented in ARPANET and PWIN) and ADCCP protocols at the IMP-IMP level of communication. The header words on a store and forward packet, other than the first one, contain bit oriented control information for the two outer levels of communication, Source IMP-Sink IMP and Host-Host. It is out of the scope of this discussion to consider these protocols.

4.3 Possible Implementation of ADCCP for ARPANET and PWIN

Assuming ARPANET and PWIN use the same overall protocol at the IMP-IMP level of communication, one can ask how ADCCP might be implemented at the level 3 graph shown in figure 4-5. Of all the ten commands and twelve responses (control words) described in ADCCP, only two are applicable here: the command SR (select and respond) and the response AC (non-delayed accept or acknowledge). For speedy transmission of packets through the network 'naks' (not accept) and delayed responses are not used. The checking of the IMP software code has become quite elaborate while the network is in operation so that a prolonged period of time for non-acceptance of a packet usually indicates a problem in the establishment of the line and shifts the IMP's attention to that possibility while the packet is routed via another link [4.8].

The framing structure for a message in ADCCP consists of the sequence F, A, C, INFO, FCS, F where

F = 8 bit flag sequence (01111110)

A = Secondary Station Link Address Field (multiple of 8 bits)

C = Control Field (8 or 16 bits) with Command/Response and Its Sequence Number

INFO = Information Field

FCS = Frame Check Sequence (a minimum of 16 bits, incremented in octets, for cyclic redundancy check)

The command SR could be used for both the store and forward packet and the routing packet while the AC response could be used for the routing packet. The address field, using two octets, could still contain the channel number of the packet in the first octet (if applicable) and the acknowledgements in the second octet. The control field, which enables one to sequentially number the control words as well as specify them need not use the numbering scheme since delayed responses and commands are not allowed. (See figure 4-6 for finite state graph.) Transparency is achieved by checking between flags for sequences of five ones and inserting a zero after each such string. A minimum of two flags frames every message and is also used for synchronization purposes.

If one examines figures 4-5 and 4-6, one immediately sees that the framing structure has been simplified by using two eight bit characters (FF) of one type rather than four eight bit characters with two sequence structures (SYN SYN DLE STX and DLE ETX SYN SYN) and four character types (SYN, DLE, STX, ETX). The interior of the packet has a single octet added for the two control words SR and AC so that the overall structure is still simpler than in the present X3.28 implementation. One would have to carefully compare the manner of implementing the transparency feature to determine which one is more efficient. Certainly, the adding of a single 0 bit in ADCCP adds less to the line traffic than an eight bit DLE character, but a crucial determining factor would be the comparative frequency of a string of five ones to a DLE character.

An overall advantage of ADCCP is that it can be separated at the outer protocol levels. As suggested in reference [4.6], the protocol in a computer network can be described via three nested levels (see figure 4-7a). The A link is the innermost IMP-IMP level and uses the format of figure 4-6. The control information for the intermediate Source IMP-Sink IMP level (the B link) would then be located at the beginning of the packet that transfers the system from states 4 to 6 in figure 4-6. Similarly, for the outermost Host-Host level (C link) the control information would appear closest to the text information in the packet. This is a last in - first out type sequencing (see figure 4-7b).

The B and C levels could take advantage of the wide range of commands appropriate for a primary/primary link and the state variables describing the sequencing of the commands and responses. With the objective of achieving standardization and efficiency, it would be very useful to examine the implementation of ADCCP at the outer levels.

4.4 Performance Comparisons of Link Control Disciplines

The two link control disciplines presently under consideration for transmitting packets on the backbone network of the future DCS, the Integrated Data Network (IDN) [2.2], will be compared in this section. The two disciplines are: (1) the character oriented protocol used for communication between the IMP's in the ARPA network [4.9], and (2) the bit oriented protocol proposed by the American National Standards Institute (ANSI) known as the Advanced Data Communications Control Procedure (ADCCP) [4.4]. The network configuration to be used as a base for the protocol comparisons is that portion of the store and forward packet switching ARPA network used for IMP to IMP communication.

4.4.1 Performance Measurements for a Communication System: The American National Standard Institute Task Group X3S3.5 identifies and defines four independent performance criteria that together measure the total performance of a system. They are: (1) The Transfer Rate of Information Bits (TRIB) that reflects the capacity of the system to handle information, (2) Transfer Overhead Time (TOT) that reflects the delay factors associated with the data communication process, (3) Residual Error Rate (RER) that reflects the accuracy or freedom from undetected errors, and (4) Availability (A) that measures the ability of a system to operate over a period of time without failure [4.10, 4.11].

Prior to developing the above performance determinations, Task Group X3S3.5 found it necessary to develop a general functional description of the system to which the performance measurements apply and to identify five possible phases of data communication, each associated with a flow of activity.

Unfortunately, not all possible systems and control procedures are applicable to the standards defined in Reference 4.10. Examples of systems and control procedures not covered by Reference 4.10 are: full duplex systems, packet switching systems, and bit oriented control procedures. Nor does the proposed standard address the total system performance for systems composed of more than one information path.

Task Group X3S3.5 is presently working on extending the applications to include, among other things, TRIB calculations for two way simultaneous links using ADCCP. The following questions under discussion, but to date still unanswered, illustrate the kind of detailed thinking required for formulating a set of standards:

(1) "How many characters should the interframe time fill contain?"

(2) "What should be the reference points of a time measurement?"

(3) "Are different performance criteria needed for the different modes of operation?"

(4) "Should transmission time include acknowledgements (explicit or implicit)?"

(5) "Is TRIB to be determined on a link or network basis?"

One can see that there still remains an enormous amount of work to be done before a set of universal performance measurement standards can be accepted. The following discussion will summarize the ANSI Task Group X3S3.5 proposed standards of 1971 with comments about their use.

4.4.1.1 System Description. The functional elements describing a system are illustrated in figure 4-8. This description can be extended to any level of communication in which imbedded levels of communication exist. One such network is the ARPA network illustrated in figure 4-9. One can possibly calculate performance measurements for each level - User to User, Host to Host, IMP to IMP, or Host to IMP. However, since the links between Host to Host and User to User are logical links, a more complex analysis than for physical links such as IMP to IMP or Host to IMP, would be required for determining performance measurements. Outer level measurements would necessarily be a function of the performance of all imbedded levels of communications. A precise measurement at any level might require data describing variables such as transmission delays, buffer capacities, error rates for all physical links and nodes, line speeds, traffic flow, length of data, etc. Other factors effecting the performance at any level are the control disciplines and software techniques for processing and queuing messages. The system description aids one in defining the end points of measurements and elements effecting the different performance criteria of a system.

The system for comparing the two modes of discipline discussed in this section is the lowest communication level in the ARPA network, IMP to IMP. Figure 4-10 illustrates the elements in this portion of the ARPA network and how they relate to the functional system elements illustrated in figure 4-8. Note that an IMP contains the communication control logic and buffers for controlling the traffic flow.

The transmission facilities of the Information Transfer Channel are synchronous, full duplex, point to point, and non-switched. The IMP to IMP link can further be described as belonging to the Primary to Primary Class 2 procedure class as described in reference 4.4. Each station acts as a primary for data initiated at its own site and as a secondary for receiving data initiated at the other site. Deferred responses are not permitted.

4.4.1.2 Phases of Data Communication. The five phases of communication defined by the ANSI Task Group X3S3.5 are as follows:

(1) Connection Establishment Phase - The information transfer channel is established during this phase.

(2) The Link Establishment Phase - The data terminal elements are identified during this phase. Supervisory functions such as polling and selecting are transmitted during this phase.

(3) The Information Transfer Phase - Messages and associated supervisory functions (i.e, acknowledgements and retransmissions of erroneous messages) are transmitted over the information path from the source data terminal to the sink data terminal.

(4) Link Termination Phase

(5) Connection Clearing Phase

Phases 1 and 5 do not apply to the subject network since it is non-switched. Phases 2 and 4 cannot be separated from phase 3 in a bit oriented protocol since the three phases are not well defined. Therefore, these phases would have to be redefined for bit oriented protocol.

4.4.1.3 Performance Criteria. The Transfer Overhead Time (TOT) measures the time required to establish and disestablish and Information Path in order to effect the information transfer. It is the ratio of the sum of the elapsed time in phases 1, 2, 4, and 5 divided by the number of information bits accepted by the receiver. Since phases 1, 2, 4, and 5 are undefined for IMP to IMP communication, TOT cannot be used as a performance criteria for comparing the two protocols discussed in this section.

The Residual Error Rate (RER) and Availability (A) involves characteristics that are generally hardware oriented - rate of undetected errors, and down time. These two performance criteria are independent of the control procedure used and will therefore not be considered in this discussion.

This leaves the Transfer Rate of Information (TRIB) as the only performance criteria for which a meaningful comparison can be made. TRIB is defined in Reference 4.10 as "The ratio of the number of information bits accepted by the receiving terminal configuration during a single Information Transfer Phase (Phase 3) to the duration of that Information Transfer Path". In defining information bits, Reference 4.10 assumes that message text and controls are composed from a subset of USASCII characters. This definition of TRIB is unacceptable for the TRIB comparison of the two disciplines to be discussed in this section. First, TRIB is defined in terms of Phase 3. It was previously shown that Phase 3 is undefined for the two disciplines. Second, the message text of the packets to be discussed later are bit oriented. The calculation of TRIB in this report can therefore not conform to any existing standards. Instead, the TRIB calculations will be based on suggestions presented in minutes of Task Group X3S3.5. Arbitrary assumptions will be made for problems still unresolved.

Mr. R. T. Moore of NBS, a member of ANSI Task Group X3S3.5, drafted a definition of TRIB on March 1, 1974, that is applicable to ADCCP. The following are excerpts from this definition:

> "TRIB is the number of Information Bits transferred on a data communication link divided by the time required for their transfer, acceptance and the availability for release of the buffer storage space they occupied for the purpose of contingent retransmission."
>
> "In a Primary - to - Primary link configuration, it is the sum of the Information Bits transmitted by a primary and received by the colocated secondary."
>
> "The time used in determining TRIB shall not include periods when interframe time fill is used in the absence of message traffic. The time required for frame retransmission, abort sequences, or inter-frame time fill for reasons other than lack of message traffic, is included in the determination of TRIB."
>
> "Information Bits are all the bits within the information field of a frame except those zeros stuffed in for flag protection. Flags, address, control octets and frame check sequence bits occur outside the information field and are not counted as Information Bits."

The effect of a control discipline on the TRIB will in large part be due to the overhead bits used in framing the packets and the added bits needed for message transparency. Error recovery procedures can also affect the TRIB value. Systems characteristics such as the error rate, speed of the line, etc., will not be considered. Nor will the variation in packet lengths be considered. An analysis of the effect of message length on the TRIB value can be found in Reference 4.12.

4.4.2 Description of the Two Control Disciplines:

4.4.2.1 Bit Oriented ADCCP. Figure 4-11a illustrates the ADCCP framing format for a transmitted packet. Since this procedure allows options to the implementer, a choice will be made wherever appropriate.

The flag sequence (F) is a sequence of 8 bits (01111110) which serves a dual purpose. F is a synchronization character and a message delimiter. Since the number of flags between packets is not a firm number, we will assume the requirement of two flags between frames. This is comparable to the two SYNs in the ARPA network.

The secondary station link address Field (A) is a variable length field N octets long ($N \geq 1$). A zero in the first bit of each octet signals the addition of another octet. A one in the first bit of an octet indicates that it is the last octet in the address field. It will be assumed for this discussion that the address field is one octet in length.

The control field (C) contains the commands, responses, and sequence numbers of the packets transmitted from one IMP to another IMP. This field too is described as N octets and N will be assumed equal to one.

The frame check sequence is similar to ARPA's the same as the 24 bit check cyclic is a redundancy check in the ARPA protocol. Since the hardware is responsible for the check, the length of FCS will be assumed equal to the cyclic redundancy check in ARPA (24 bits).

Transparency in ADCCP is accomplished by inserting a zero bit following five contiguous one bits anywhere between the beginning and ending flags of the frame.

All sequentially numbered command frames must be transmitted and received in sequence. The primary expects an acknowledgement for all sequentially numbered commands. State variables are used by the receiver to keep a record of the accepted command frames requiring acknowledgement. If an acknowledgement is received by the Primary for command sequence N, then it is assumed that all previous numbered command frames are acknowledged. If a command frame is in error, the exception state variable (E) is set and all succeeding command frames are discarded. All response frames will be assumed to contain no information field.

4.4.2.2 <u>Character Oriented IMP to IMP ARPA Control Procedure</u>. The ARPA protocol requires two SYN characters between frames. The message delimiters (DLE STX and DLE ETX) are each two eight bit characters. The end of Test (DLE ETX) delimiter is followed by a 24 bit cyclic redundancy check (CC1, CC2, CC3). (See figure 4-11b.)

A five sixteen bit word header is tacked on to the original packet by the source IMP before the packet is routed to the destination Host. The first word contains the equivalent of an address and sequence number for the neighboring IMP. The channel number with its associated odd/even bit field can be compared to the sequence number in the ADCCP protocol. The channel number is assigned by the sending IMP and the corresponding acknowledge bits are identified by the receiving IMP. The first work is peeled off by the receiving IMP and replaced with a similar word before advancing the packet to the next IMP in the route.

The remaining four words contain, essentially, addresses, commands, responses, and sequence numbers for upper level communication. Fields such as priority and routing, will be ignored in this discussion. All upper level communication control words will be considered part of the information packet transmitted between IMPs. This assumption implies that an equal number of words are used for upper level communication in the ADCCP protocol. The heading field in figure 4-11b is thus reduced to 16 bits and the maximum size information field is increased to 67 words.

Transparency is provided by inserting a DLE character after all DLE characters that appear in the heading or packet.

Only one command and one response are used in the ARPA protocol. The command SELECT AND RESPOND is implied whenever a packet is transmitted. A negative acknowledgement is also implied after a given interval of time has elapsed and no positive acknowledgement has been received by the sender. The sender after an implied negative acknowledgement (no response) retransmits the packet. Packets are individually acknowledged by the receiver by storing a one in the appropriate acknowledgement bit of the header word. Eight packets can be acknowledged in one frame, and it can be "piggy-backed" on a command frame. Acknowledgements do not have to be in sequence. The queuing technique used for storing the packets enables the sender to retransmit a single packet that may be out of sequence.

4.4.3 Performance Comparisons:

It would be unrealistic at this time to compare TRIB values for an implemented protocol (ARPA) and a theoretical protocol (ADCCP), using data transmissions for which one would have to make unwarranted assumptions. Thus, no assumptions will be made with respect to the start-up procedure. The time interval in each example will be for the transmission of eight maximum length packets. ARPA cannot transmit more than eight packets without an acknowledgement.

The examples in figures 4-12 through 4-15 compare the TRIB values of the protocols for normal transmission, an error in the transmission of a packet, an error in one of the acknowledgements, and for achieving transparency. It is beyond the scope of this report to demonstrate the immediate negative responses available in ADCCP in contrast to ARPA's time-out. Nor will an attempt be made to estimate the probability of a specific bit configuration occurring in a packet (i.e., 01111110 or DLE).

4.4.3.1 Assumptions. The following will be assumed for both protocols:

(1) The number of information bits per packet is 1072.

(2) Overhead bits due to the address, sequence number and command/response in the header equals 16 bits.

(3) The cyclic redundancy check equals 24 bits.

(4) The transmission time interval will start with the first of the two interframe synchronization characters and stop with the second interframe flag.

(5) The time for transmitting two characters will be allowed for the propagation delay of a packet and the secondary station reaction time.

(6) Only one station is transmitting packets.

(7) The command is SELECT and RESPOND.

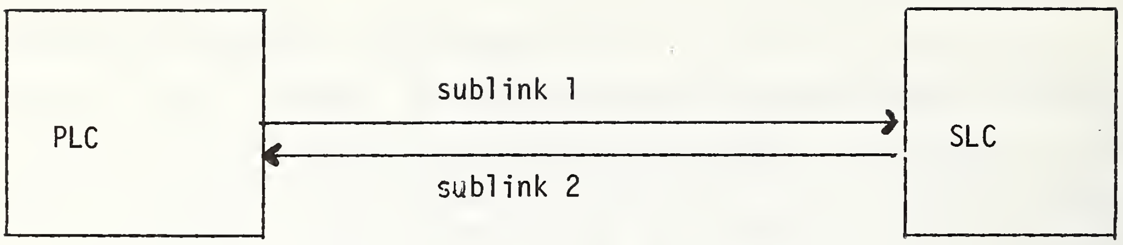
Additional assumptions made for the ARPA protocol are: (1) Only one acknowledgement is sent for all eight packets and it is sent in a "null" packet of 16 bits, (2) The time-out before a packet is retransmitted equals the time to transmit and acknowledge eight maximum size packets.

4.4.3.2 <u>Examples</u>. The following defines the notations used in the examples in figure 4-12 through 4-15:

Ci ↔	The command header for transmitting packet i.
Ri ↔	A positive acknowledgement for command i.
NRi ↔	A negative acknowledgement for command i.
$P \rightarrow S$	Means transmission from the primary station to the secondary station.
$S \rightarrow P$	Means transmission from the secondary station to the primary station.
X	Indicates an error.

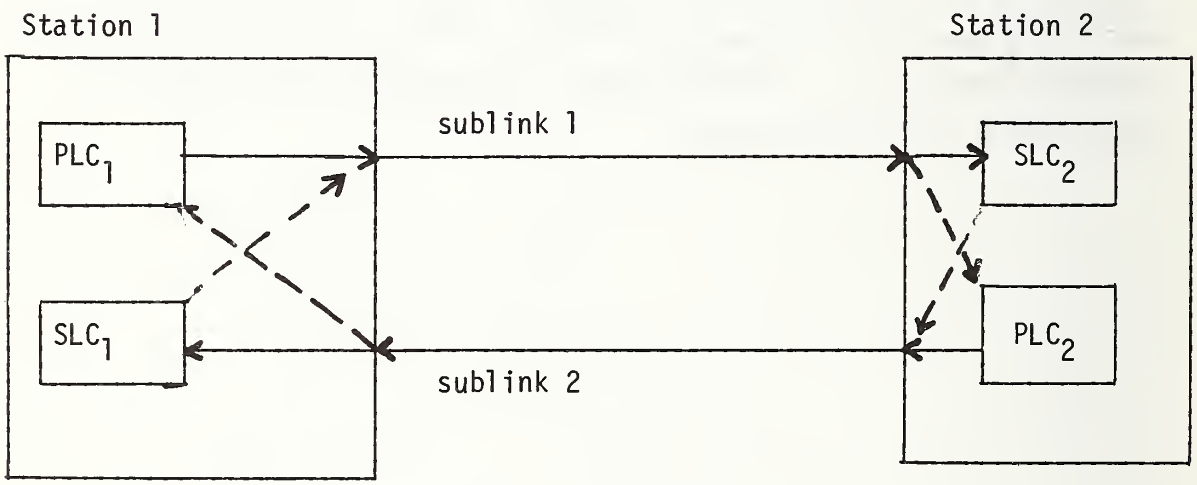
4.5 Conclusions

The preceding analyses are not, by any means, a complete evaluation of ADCCP for packet-switching computer networks. However, they do provide initial insights into comparative complexity and performance of the bit oriented technique versus the character oriented bisynchronous discipline. The results tend to establish the feasibility of ADCCP as a potential future discipline for comupter networks. A number of assumptions had to be made here to establish meaningful comparisons in straightforward analysis. Less restrictive assumptions and more elaborate analysis - even computer simulations - should be undertaken, to reflect the complexity of real applications and yield more precise comparisons. The impact of ADCCP on the realization of higher level protocols - e.g. the flow control and buffer allocation activity in ARPANET - should also be pursued in depth. The potential for much commercial activity in implementing ADCCP computer interfaces is already evident, and heightens the engineering significance of this discipline.



Closed Path: PLC ⟶ sublink 1 ⟶ SLC ⟶ sublink 2 ⟶ PLC

FIGURE 4-1a. Primary/Secondary Link Model Two-Way Alternate.



Closed Paths: PLC_1 ⟶ sublink 1 ⟶ SLC_2 ⟶ sublink 2 ⟶ PLC_1 or

or PLC_2 ⟶ sublink 2 ⟶ SLC_1 ⟶ sublink 1 ⟶ PLC_2

FIGURE 4-1b. Primary/Primary Link Model Two-Way Alternate or Simultaneous.

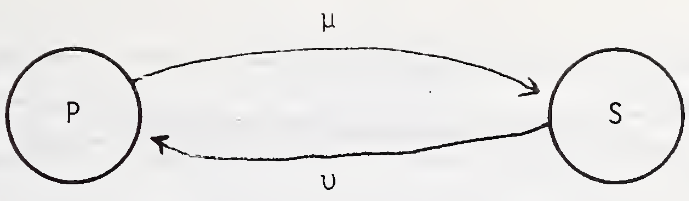


FIGURE 4-2a. Finite State Graph for Figure 4-1a: Level 1

Alphabet→ State↓	μ	υ
P	S	E
S	E	P

FIGURE 4-2b. Finite State Matrix for Figure 4-1a: Level 1

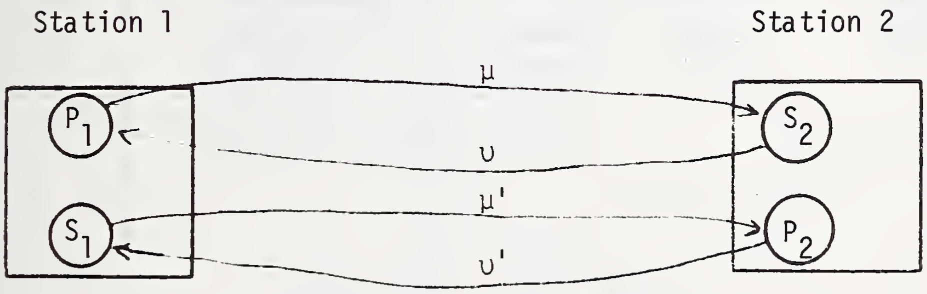


FIGURE 4-3a. Finite State Graph for Figure 1b: Level 1
Two-Way Alternate

Alphabet→ State↓	μ	υ	μ'	υ'
P_1	S_2	E	E	E
S_2	E	P_1	E	E
P_2	E	E	S_1	E
S_1	E	E	E	P_2

FIGURE 4-3b. Finite State Matrix for Figure 1b: Level 1
Two-Way Alternate

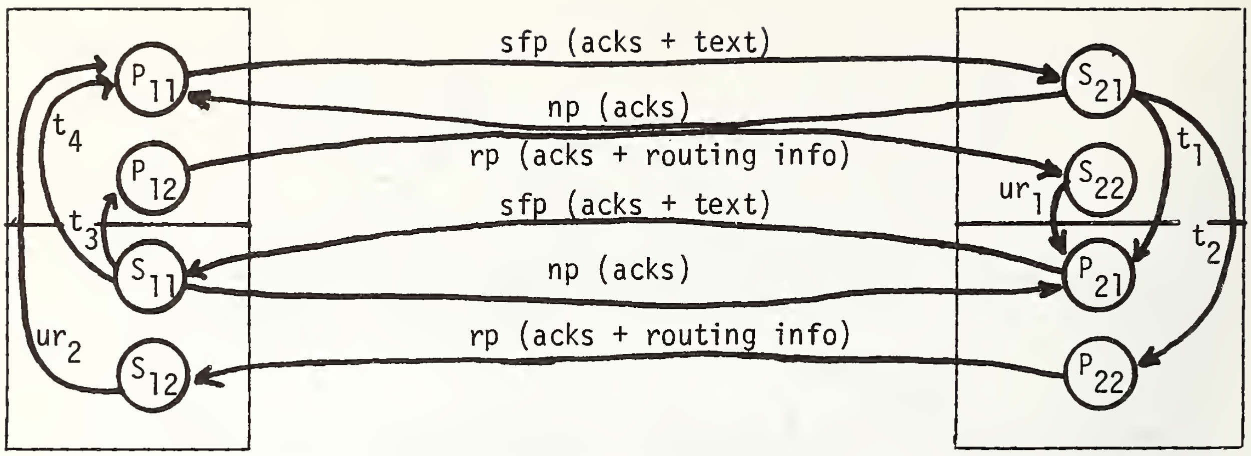


FIGURE 4-4a. Finite State Graph for Figure 1b: Level 2 Two-Way Alternate.

Alphabet State	Store and Forward Packet (sfp)	Routing Packet (rp)	Null Packet (np)	Software Actions Denoted by Transitions t_1, t_2, t_3, t_4	Software Actions Updating Routing ur_1, ur_2
P_{11}	$S_{21}, (t_1, t_2)$			(P_{21}, P_{22})	
P_{12}		$S_{22}, (ur_1)$			
S_{11}			P_{21}		
S_{12}					(P_{11})
S_{21}			P_{11}		
S_{22}					(P_{21})
P_{21}	$S_{11}, (t_3, t_4)$			(P_{12}, P_{11})	
P_{22}		$S_{12}(ur_2)$			

FIGURE 4-4b. Finite State Matrix for Figure 1b: Level 2 Two-Way Alternate.

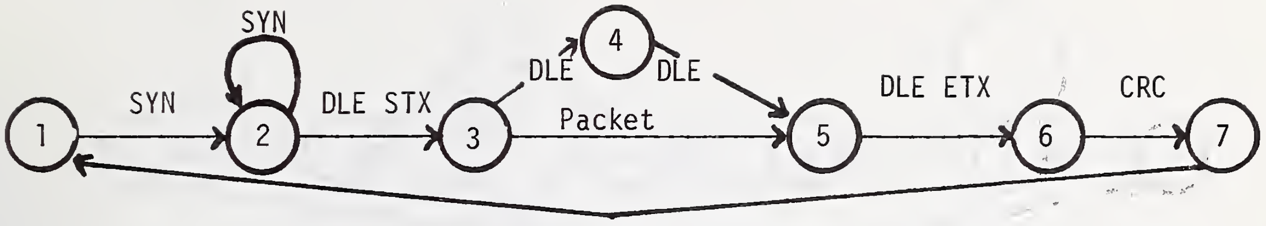


FIGURE 4-5a. Detailed Picture of Packet Transmission Using X3.28: Level 3

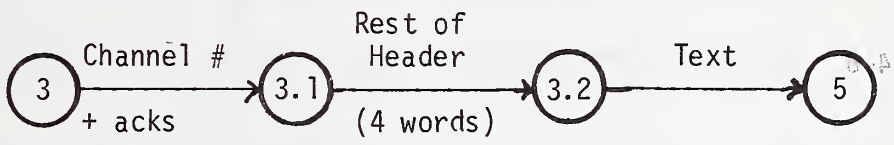


FIGURE 4-5b. Explosion of Store and Forward Packet

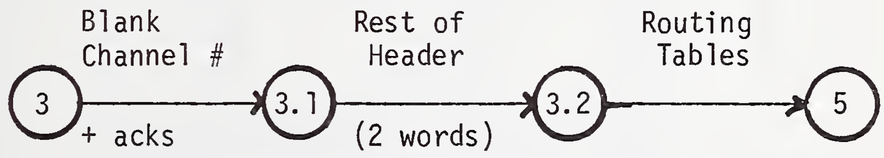


FIGURE 4-5c. Explosion of Routing Packet

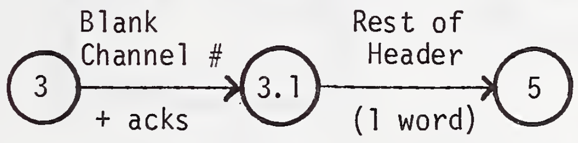


FIGURE 4-5d. Explosion of Null Packet

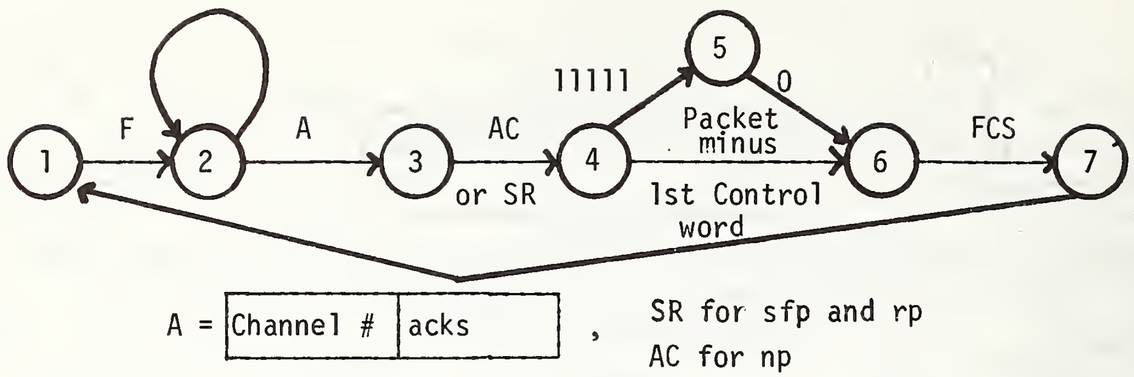
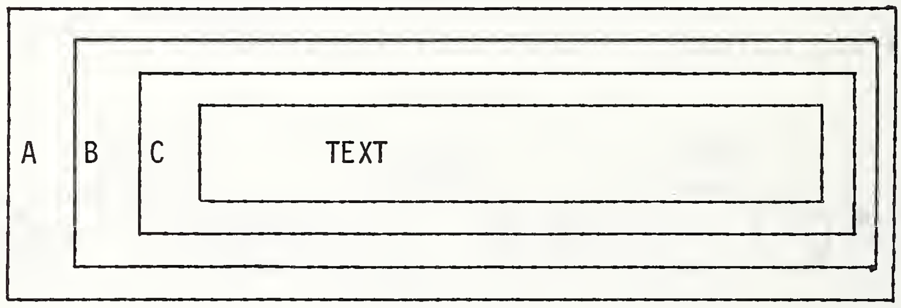


FIGURE 4-6. ADCCP Implementation of Packet Transmission: Level 3



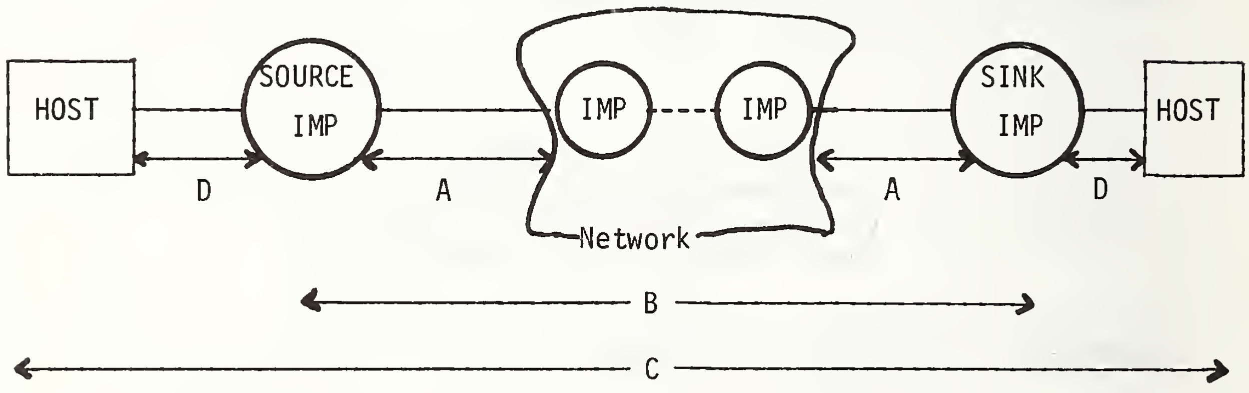


FIGURE 4-7a. Protocol Levels in Computer Network

Link A		Link B		Link C		
A	C	A	C	A	C	TEXT

FIGURE 4-7b. Structure of Packet with Control Words for Three Nested Levels of Protocol

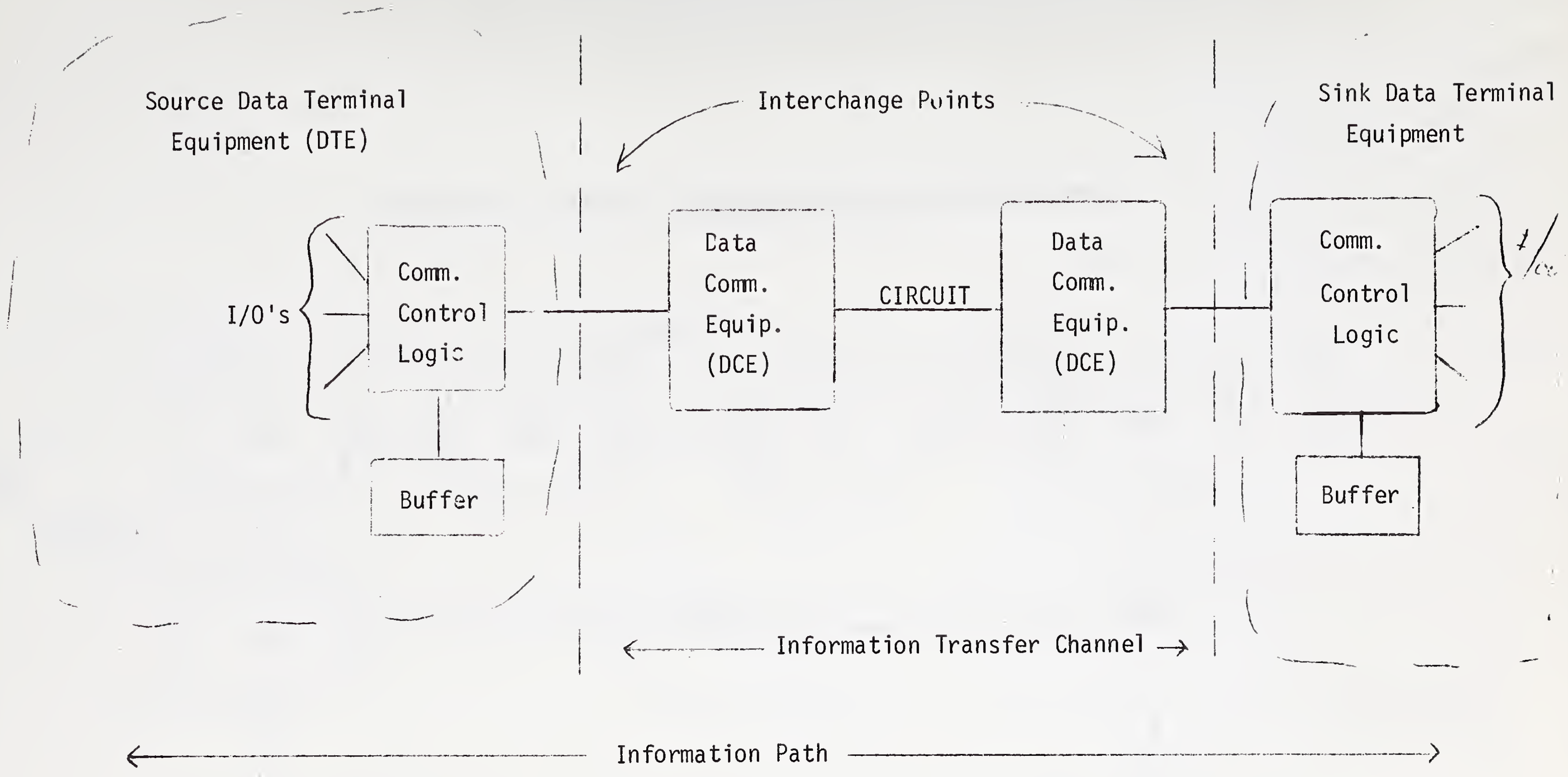


FIGURE 4-8. Functional Elements of a Communication System.

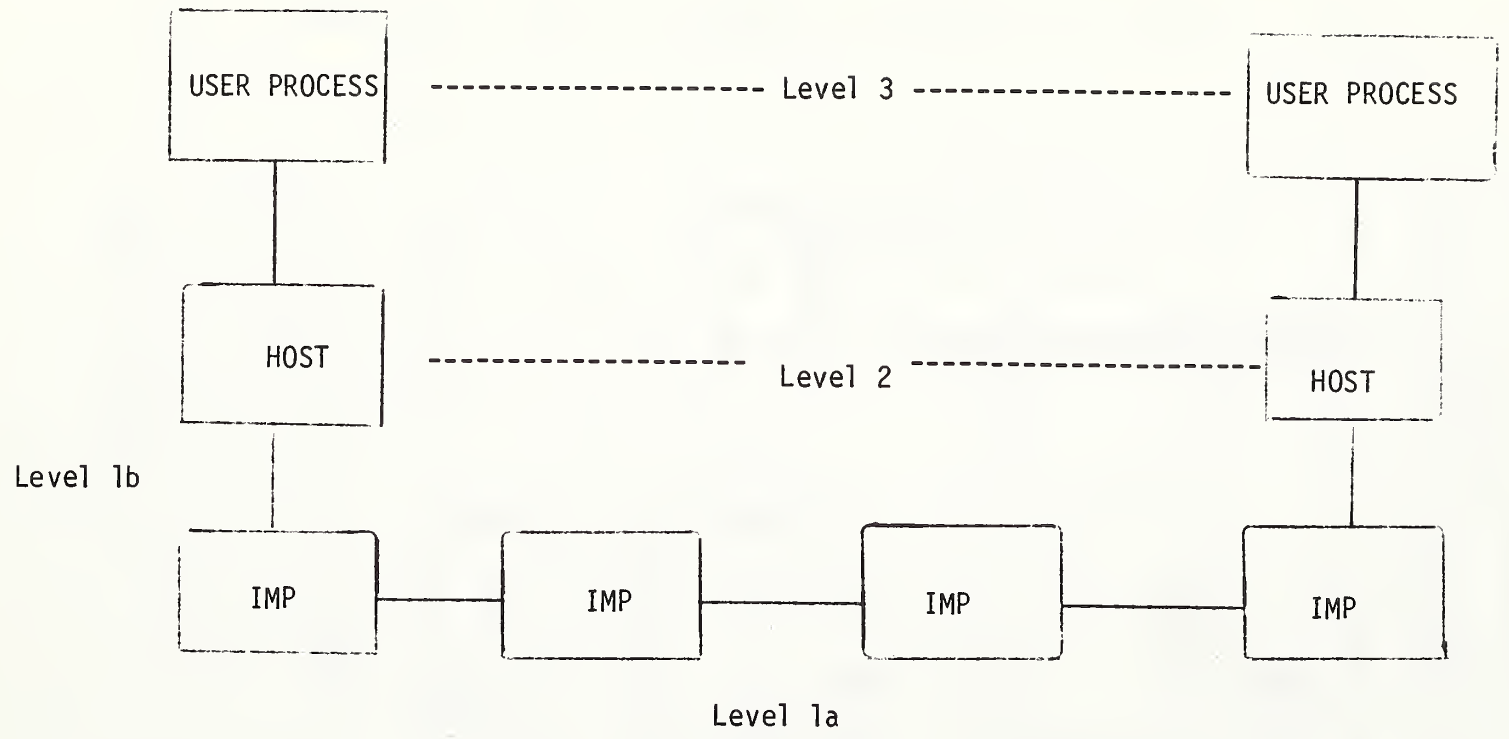


FIGURE 4-9. Levels of Communication in the ARPA Network.

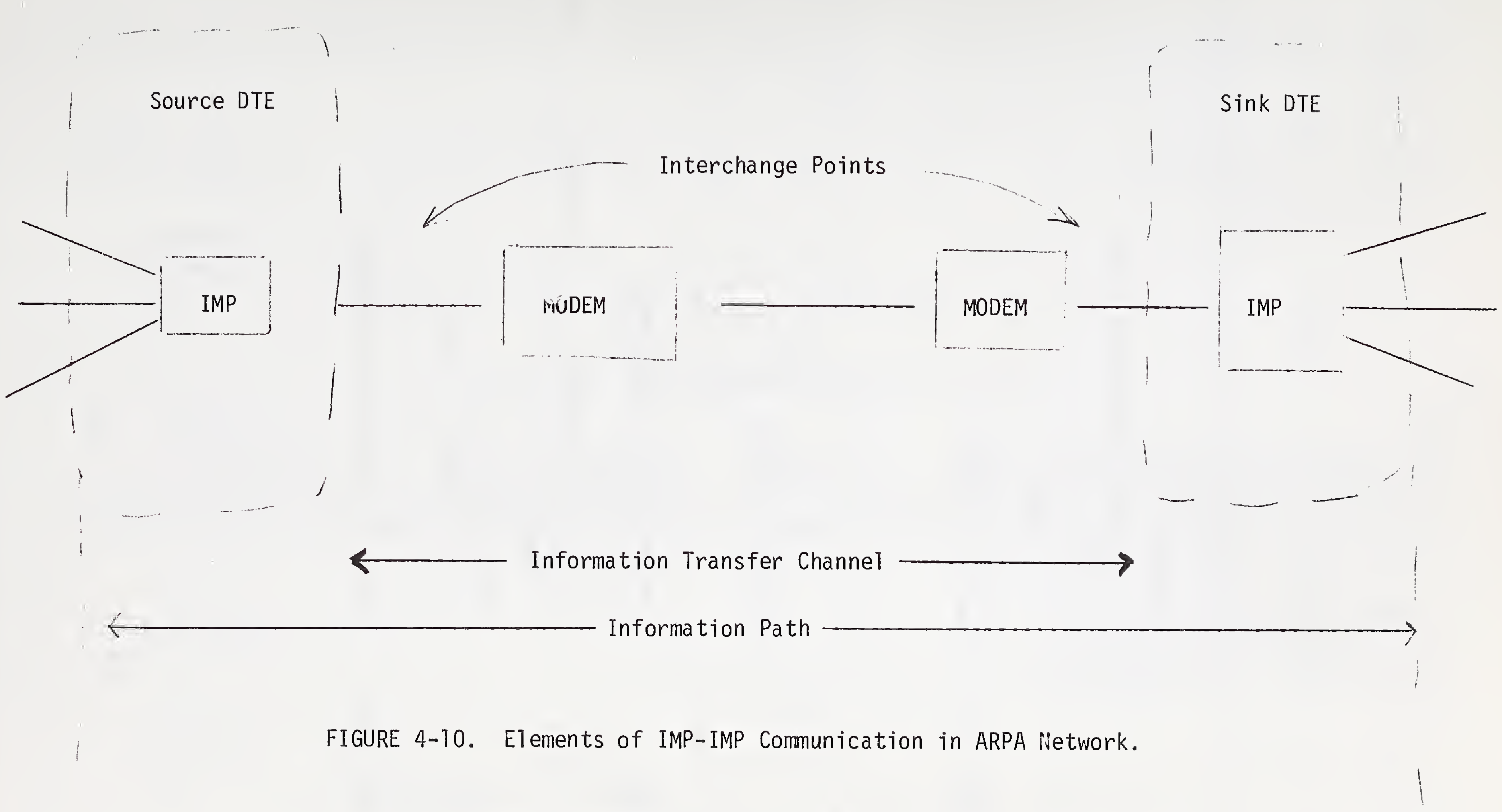


FIGURE 4-10. Elements of IMP-IMP Communication in ARPA Network.

F	A	C	PACKET	FCS	F

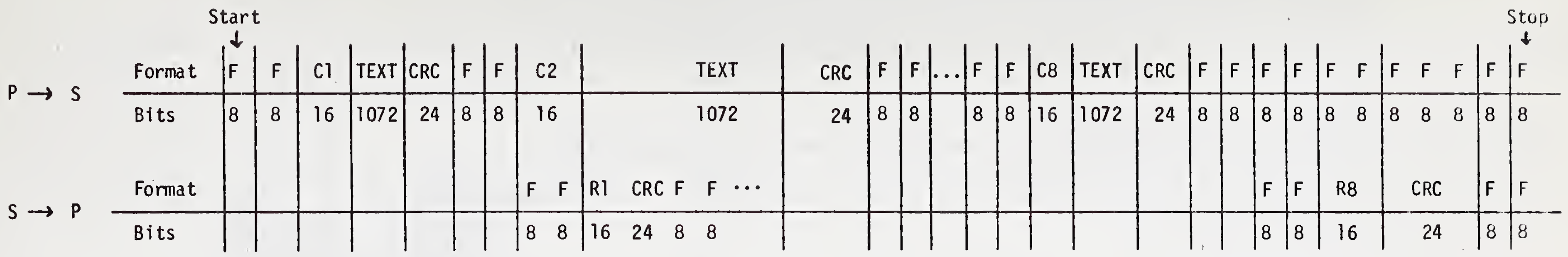
F	=	Flag Sequence
A	=	Secondary Station Link Address Field
C	=	Control Field
FCS	=	Frame Check Sequence

FIGURE 4-11a. Framing for ADCCP.

S Y N	S Y N	DLE STX	Heading	PACKET	DLE ETX	CC1	CC2	CC3	S Y N	S Y N

SYN	Synchronization Character
DLE STX	Start of Text
Heading	Five words of commands and addresses for all levels of communication
DLE ETX	End of Text
CC1, 2, 3	Cyclic redundancy check

FIGURE 4-11b. Framing for IMP-IMP ARPA Protocol.

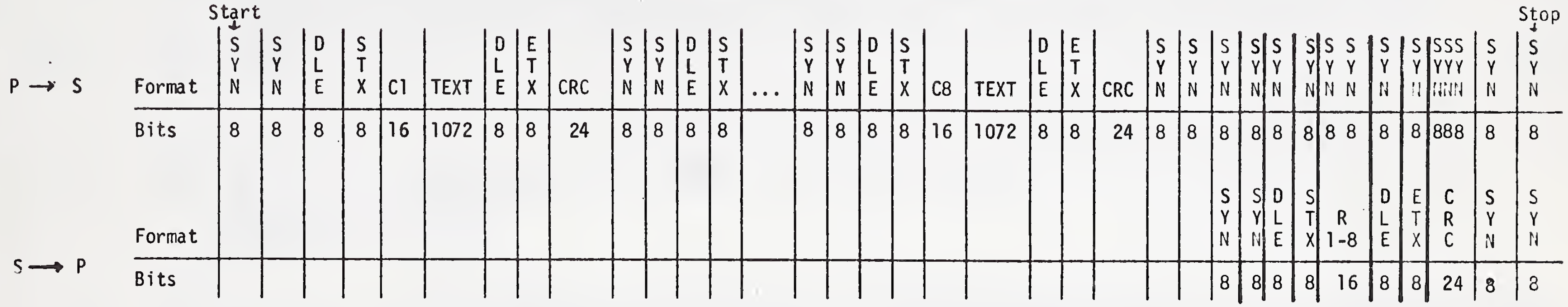


Information Bits = 8576 bits

Bit Periods = 9112 bits

TRIB $= \frac{\text{Information Bits}}{\text{Bit Periods}}$ X BPS $= \frac{8576}{9112}$ X BPS = .941 x BPS bits/sec

FIGURE 4-12a. Normal Transmission Using ADCCP Protocol.

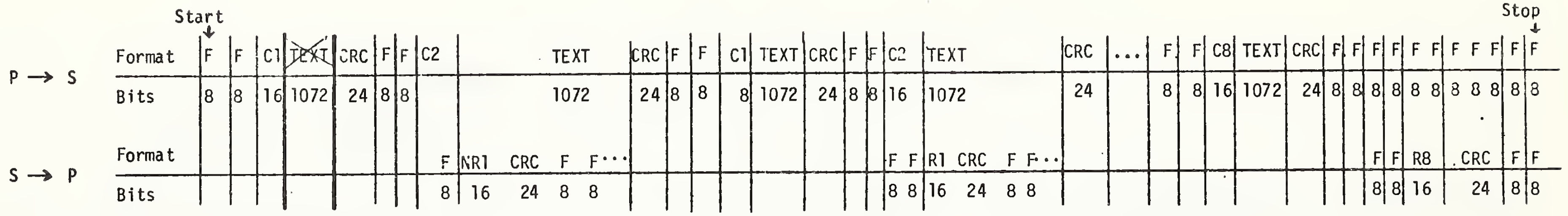

Start
Stop
P → S
Format
SYN SYN DLE STX C1 TEXT DLE ETX CRC SYN SYN DLE STX ... SYN SYN DLE STX C8 TEXT DLE ETX CRC SYN SYN SYN SYN SYN SYN SYN SYN SYN SSS YYY NNN SYN SYN
Bits
8 8 8 8 16 1072 8 8 24 8 8 8 8 8 8 8 8 16 1072 8 8 24 8 8 8 8 8 8 8 8 8 8 888 8 8
S → P
Format
SYN SYN DLE STX R 1-8 DLE ETX CRC SYN SYN
Bits
8 8 8 8 16 8 8 24 8 8

Information Bits = 8576 bits

Bit Periods = 9400 bits

TRIB $= \frac{\text{Information Bits}}{\text{Bit Periods}}$ X BPS $= \frac{8576}{9400}$ x BPS = .912 X BPS bits/sec

FIGURE 4-12b. Normal Transmission Using ARPA Protocol.

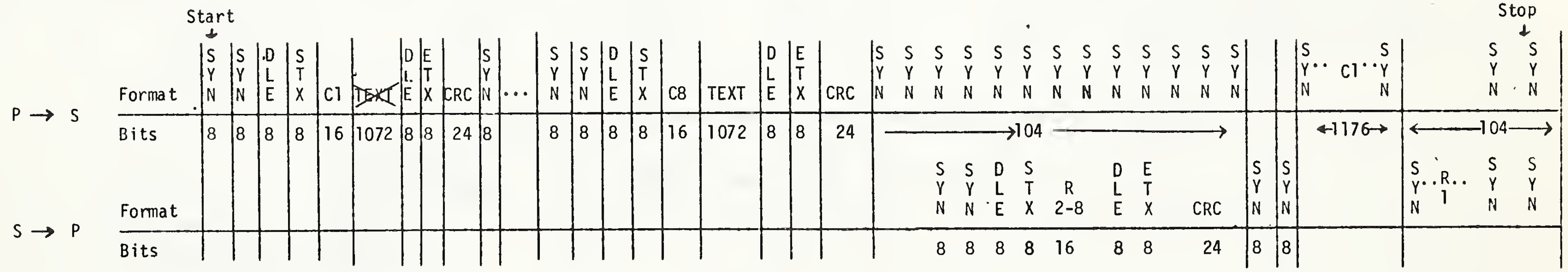


Information Bits = 8576 bits

Bit Periods = 11368 bits

TRIB $= \frac{\text{Information Bits}}{\text{Bit Periods}} \times \text{BPS} = \frac{8576}{11368} \times \text{BPS} = .754 \times \text{BPS}$ bits/sec

FIGURE 4-13a. Error in Text of First Packet - ADCCP Protocol.

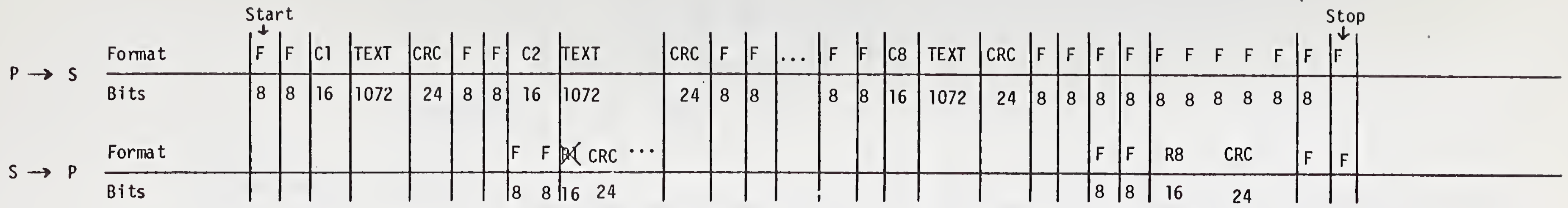

Start
Stop
P → S
S → P
Format
Bits
SYN SYN DLE STX C1 TEXT DLE ETX CRC SYN ... SYN SYN DLE STX C8 TEXT DLE ETX CRC SYN SYN SYN SYN SYN SYN SYN SYN SYN SYN SYN SYN SYN SYN.. C1..SYN SYN SYN
8 8 8 8 16 1072 8 8 24 8 8 8 8 8 16 1072 8 8 24
104
1176
104
SYN SYN DLE STX R 2-8 DLE ETX CRC SYN SYN SYN..R..1 SYN SYN
8 8 8 8 16 8 8 24 8 8

Information Bits = 8576 bits

Bit Periods = 10680 bits

TRIB $= \frac{\text{Information Bits}}{\text{Bit Periods}} \times \text{BPS} = \frac{8576}{10680} \times \text{BPS} = .803 \times \text{BPS}$ bits/sec

FIGURE 4-13b. Error in Text of First Packet - ARPA Protocol.



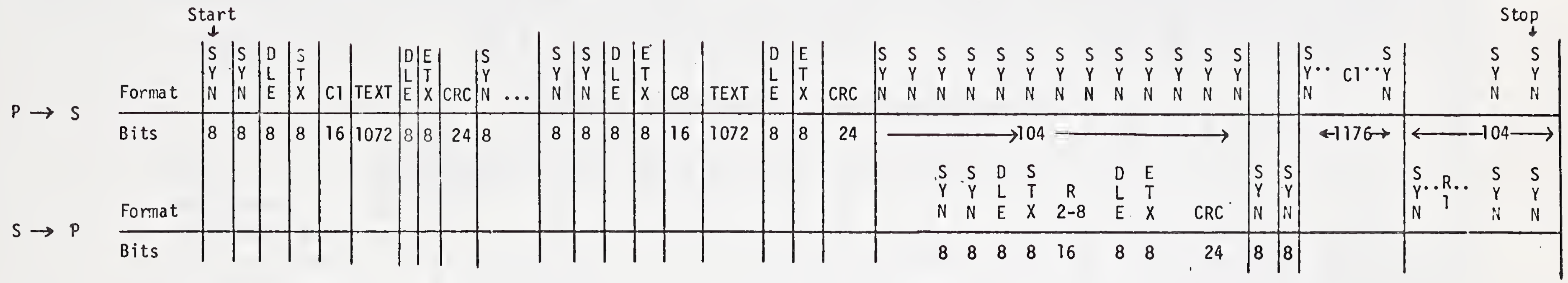
Information Bits = 8576 bits

Bit Periods = 9112 bits

TRIB $= \dfrac{\text{Information Bits}}{\text{Bit Periods}} \times \text{BPS} = \dfrac{8576}{9112} \times \text{BPS} = .941 \times \text{BPS bits/sec}$

Note: Rate is not affected by the error.

FIGURE 4-14a. Error in Response - ADCCP Protocol.

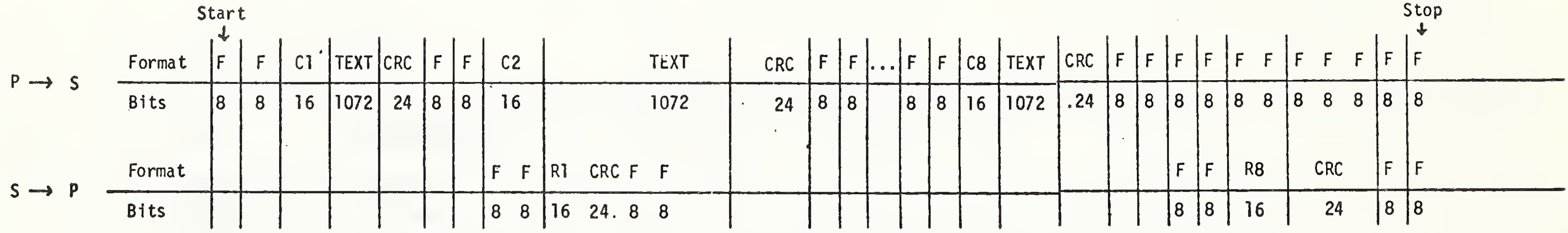

P → S
S → P
Format
Bits
Start
Stop
SYN SYN DLE STX C1 TEXT DLE ETX CRC SYN ... SYN SYN DLE ETX C8 TEXT DLE ETX CRC SYN SYN SYN SYN SYN SYN SYN SYN SYN SYN SYN SYN SYN SYN..C1..SYN SYN SYN
8 8 8 8 16 1072 8 8 24 8 8 8 8 8 16 1072 8 8 24
104
1176
104
SYN SYN DLE STX R 2-8 DLE ETX CRC SYN SYN SYN..R 1.. SYN SYN
8 8 8 8 16 8 8 24 8 8

Information Bits = 8576 bits

Bit Periods = 10680 bits

TRIB $= \dfrac{\text{Information Bits}}{\text{Bit Periods}} \times \text{BPS} = \dfrac{8576}{10680} \times \text{BPS} = .803 \times \text{BPS bits/sec}$

FIGURE 4-14b. Error in Response of First Packet - ARPA Protocol.



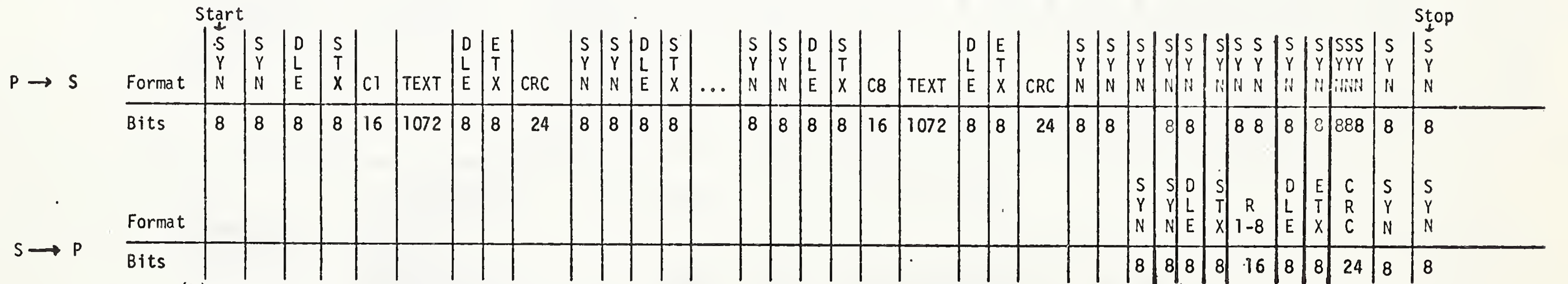
(a) ADCCP Protocol - assume that each packet contains one light bit configuration of 01111110. A zero is inserted after the fifth 1.

Information Bits = 8576 bits

Bit Periods = 9120 bits

TRIB $= \frac{\text{Information Bits}}{\text{Bit Periods}}$ X BPS = $\frac{8576}{9120}$ X BPS = .940 x BPS bits/sec

FIGURE 4-15a. ADCCP Protocol - Transparency Comparisons.



Start
Stop
P → S
S → P
Format
Bits
SYN SYN DLE STX C1 TEXT DLE ETX CRC SYN SYN DLE STX ... SYN SYN DLE STX C8 TEXT DLE ETX CRC SYN SYN SYN SYN SYN SYN SYN SYN SYN SYN SYN SSS YYY NNN SYN SYN
8 8 8 8 16 1072 8 8 24 8 8 8 8 8 8 8 8 16 1072 8 8 24 8 8 8 8 8 8 8 8 888 8 8
SYN SYN DLE STX R 1-8 DLE ETX CRC SYN SYN
8 8 8 8 16 8 8 24 8 8

(b) ARPA Protocol - assume that each packet contains one coded character equivalent to DLE. Another DLE is inserted after the DLE.

Information Bits = 8576 bits

Bit Periods = 9464 bits

TRIB $= \frac{\text{Information Bits}}{\text{Bit Periods}}$ X BPS = $\frac{8576}{9464}$ x BPS = .906 X BPS bits/sec

FIGURE 4-15b. ARPA Protocol.

5. FEASIBILITY OF COMMON COMMUNICATIONS SOFTWARE

For the near future, WWMCCS networking will apparently depend upon a variety of communications interfaces and disciplines as suggested in figure 2-1. Moreover, within the HIS 6000, a number of software components are involved with these interfaces and the associated applications modes that may have unique communications conventions - e.g. GRTS-6000, NCP software for PWIN, TPE, Direct Access, etc. A concern for optimization and high performance within individual communications modes would argue for separate interface processors and software, tailored uniquely to the control actions, buffer sizes, and timing criteria of each. On the other hand, the high cost of this specialized approach, both in original software development and in continuing consumption of host computer resources, suggests that a single interface processor with one common software approach, could be more economical and effective overall. Of course, a feasibility study is necessary to determine what software approach could meet this range of requirements and whether adequate performance and configuration flexibility could be achieved. If proven feasible through analysis and prototype implementation, this concept could lead to networking standards in terms of the functional interface between the host computer and the interface processor, and also the functional requirements for the interface processor itself.

As a first step in such a feasibility study, this project has examined the design structure of communications processing software in relation to the requirements of different disciplines. It is concluded that a uniform software approach is conceptually feasible, and further steps in the feasibility study are suggested.

5.1 Inherent Processing Requirements

Briefly stated, a computer-communications interface processor is a device for buffering, control, and conversion between different communications processes. In the simplest case, only two line types are involved: to the host computer, and to the remote terminals (including computers). However, experience shows that generally one may need to accommodate several types of line procedures, whether to local computers or remote terminals, within a computer network. Assuming the processor is implemented via a programmable minicomputer, the state-of-the-art design approach between hardware and software is depicted in figure 5-1. This diagram represents what may be called an interrupt-driven, task-oriented software concept, reflecting the inherent processing applicable to <u>any</u> communications discipline. Timing, sensing, assembly and error detection at the bit level on individual lines would of course be implemented in the hardware line interface for each type (one or more logic cards, for example). Line interface hardware could be reconfigured in the field to accommodate the needed number of each type.

Software tasks include implementing the control procedure (e.g. ANSI X3.28 or ADCCP) for each line, managing processor storage, editing of messages or frames to recover control information, routing, etc. as already discussed in Section 3. The first issue is what approach to task scheduling and management would be pertinent to serving a variety of communications disciplines?

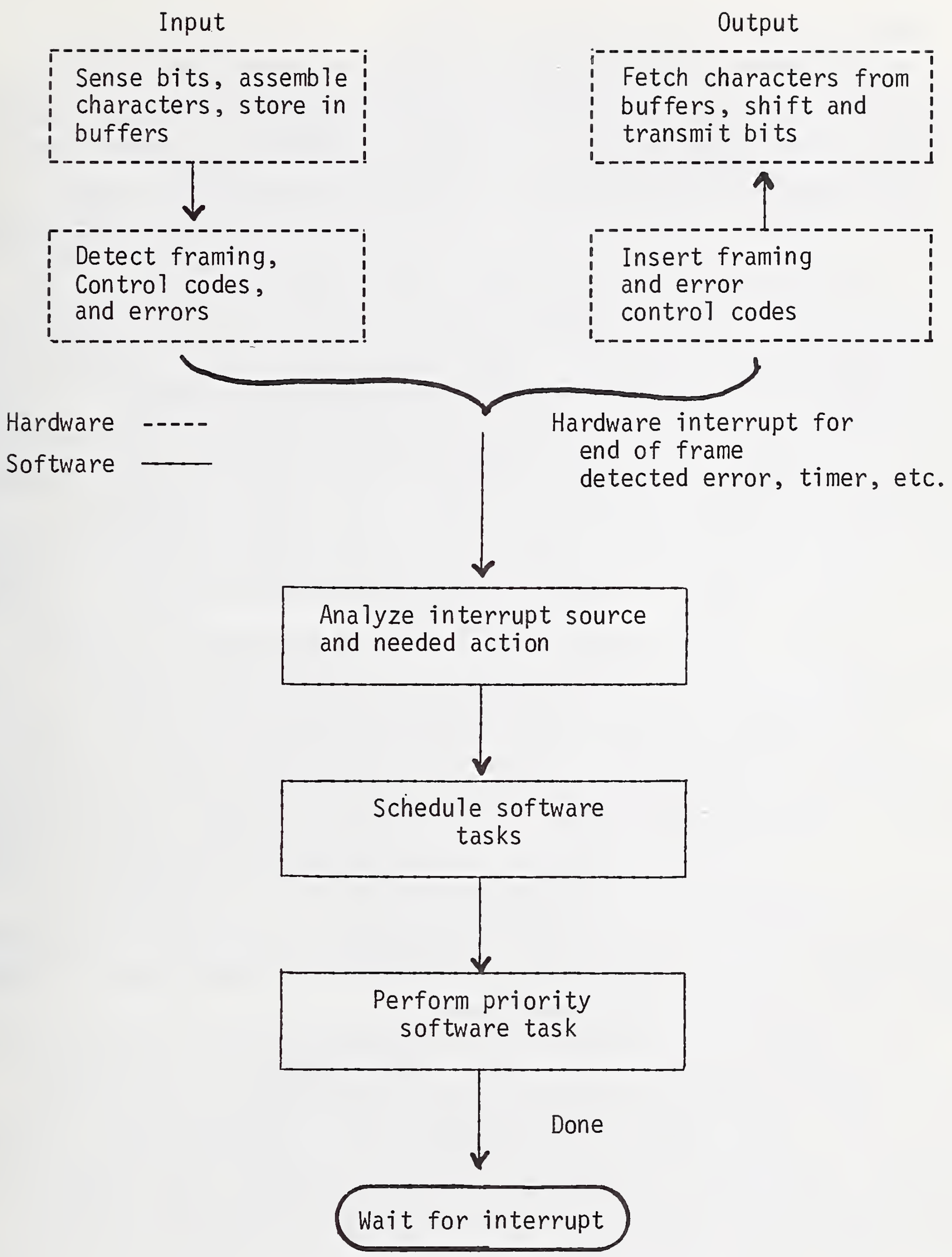

Input
Output
Sense bits, assemble characters, store in buffers
Fetch characters from buffers, shift and transmit bits
Detect framing, Control codes, and errors
Insert framing and error control codes
Hardware -----
Software ——
Hardware interrupt for end of frame detected error, timer, etc.
Analyze interrupt source and needed action
Schedule software tasks
Perform priority software task
Done
Wait for interrupt

FIGURE 5-1. Basic Hardware/Software Processing for Communications.

5.2 Design Approach Communications Software

The development of communication software is a costly and complex undertaking; the minimization of that cost and complexity leads to a consideration of common communication software and system expandability. Common communication software is a collection of programs which are shared among different processes. Initially the processes are divided into subprocesses of which as many as possible are common to more than one process. The processes, when expressed as programs, then take the form of a collection of subprograms which are shared among the processes. This idea-common software- is not new since the concept of subprograms has long been used, but the exploration of system characteristics to maximize the use of common software still needs to be done. Related to the purpose of common software is the idea of system expandability which is the concept of adding new processes to an existing system with little but preferably no modifications to the existing software. Since the communications processes of some systems can be augmented only by complex modifications of the operating system, expandability is an important requirement for communications systems.

Thus the characteristics of the software system that simplify the implementation of both common software and expandability deserve to be considered. The characteristics of any software system are based on the structure of its operating system. A communication operating system differs from a more general one only in the degree to which some features are emphasized and some are not. The real time response and input-output rates are emphasized while features such as higher level languages (e.g. FORTRAN, ALGOL) or text editor are not emphasized.

One of the primary functions of an operating system is to rebuild a computer that is non-deterministic due to cycle stealing and interrupts into a more or less deterministic automaton. This reconstruction is necessary to prevent system errors, to provide a means of proving the correctness of the software, and to provide structure for implementing common software and expandability.

One method for taming the degree of indeterminancy is a layered operating system. This approach, fostered by Dijkstra [5.4], consists of a hierarchical structure of system functions each communicating to only the adjacent and dependent only on the lower layers and itself for its correct functioning.

The nucleus of the system should have features to regulate the interaction of multiple processes. This regulation can be further classified into features of control and communication. The control of processes involves creation, scheduling, and removal of processes. Creation of a process involves the allocation of system resources to that process. For internal processes - those involving programs - core storage is allocated. External processes - input/output operations - require the allocation of peripheral devices upon creation. A name by which the process is later referenced is associated with it upon creation.

Scheduling of processes has a policy and implementation aspect. The policy of scheduling is an algorithm for deciding which processes are to be activated, which are to be stopped, and which are to be interrupted. The algorithm for scheduling belongs outside the nucleus while the mechanisms of implementing the algorithm belong in the nucleus. These mechanisms are the starting, stopping, and suspension of a process.

After the creation of a process, the process is ready to be started. The start operation may be implemented in two ways; one, the start operation places the process name in a queue to be served by the next available processor; two, the start operation causes the system to assign a priority to the process after which it is placed in a queue of processes awaiting a processor.

While the start operation is simple, the stop operation is more involved since the process may have created other processes. Thus the process to be stopped, the parent process, may have child processes which must be stopped before the parent process is stopped; so the stop operation may have to be iterated along a chain of processes beginning with the parent.

The operating system previously mentioned, a multiprogrammed operating system, has a vital requirement that it allow the system manager to change the mode of operation it controls; for otherwise his freedom to add new communication functions to the system can be seriously limited. Unfortunately most currently used operating systems do not allow such a change. Most are based on a non-hierarchical structure of system functions so that they operate in a single mode such as real time, batch processing, priority scheduling, or time slicing. In addition, changes to a system which has non-hierarchical structure is a complex undertaking.

When the need arises, the manager often finds it hopeless to modify an operating system designed on rigid assumptions about a specific mode of operation. The alternative - to replace the operating system with a new one - is also an equally if not a more difficult matter because the other software is tightly tied to interface conventions of the original operating system. The main problem in the design of a multiprogrammed operating system is not to define functions that satisfy specific operational requirements, but rather to supply a system nucleus that can be extended in an orderly manner.

The purpose of system nucleus is to implement the following fundamental concepts: process, interprocess communication, and process control. A process, which often is identified with a program, is more generally the execution of one or more programs or an input-output operation. The former are called internal processes while the latter are called external operations.

The distinction between internal and external processes, based on differences in process scheduling and storage addressing has a drastic influence on the real time and expandability characteristics of the system. On the one hand, input-output operations can be indicated almost immediately by interrupts and run without preemption for several milliseconds. On the other hand, due to a fixed type of scheduling, internal processes could only respond to urgent external events in 10 to 100 milliseconds. The modification of any input-output process often requires reassembly and testing of the entire operating system, while internal processes are comparatively easy to implement and test.

Where there are two or more processes that interact or exchange information, mechanisms must be provided for the synchronization of these processes. Synchronization is necessary to prevent multiple processes from interfering with the transfer of information between them, such as may occur when some processes write information into a buffer from which others read the information. The mechanisms available for synchronization are Dijkstra's lock and unlock operations operating on semiphores [5.1], Hansen's condition critical regions [5.2], and his communication primitives [5.3]. The latter, which implicitly uses Dijkstra's operations, avoids the problems of the programmer error of misusing Dijkstra's operations because those operations are part of the system nucleus normally unavailable to the programmer.

The following are Hansen's primitives:

SEND MESSAGE (RECEIVER, MESSAGE, BUFFER)
WAIT MESSAGE (SENDER, MESSAGE, BUFFER)
SEND ANSWER (RESULT, ANSWER, BUFFER)
WAIT ANSWER (RESULT, ANSWER, BUFFER)

With these primitives additional processes could more easily be added to the system than with a convention that allows interprocess communication by direct transfers. Because the communication is checked by the nucleus, errors affect only the process that initiates the communication. Thus it is possible, with hardware memory protection, to check out a new process while the system is operating on the normal workload.

In the above, the parameter message refers to the location of the message to be received or sent, buffer is the location of a system supplied buffer area to hold either the message or the answer, and answer is the location of the answer. As implemented on the RC4000 the primitives function in the following manner.

Send Message copies a message into the next available buffer and puts the buffer into the queue of the named receiver. The receiver is activated if it is waiting for a message (if it has issued a wait message command) while the sender, after having received the address of the buffer, continues with is process.

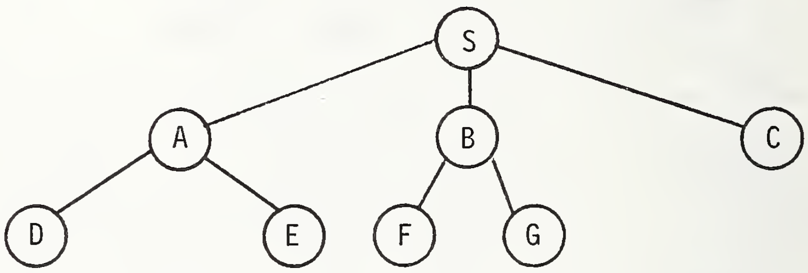
Wait Message delays the requesting process until a message arrives in its queue. When the process is allowed to proceed, it is supplied with the name of the sender, the contents of the message, and the address of the message buffer. The buffer is then removed from the queue and held for an answer.

Send Answer puts an answer into the buffer in which some preceeding message was received and then puts the buffer in the queue of the original sender. The sender is activated if it is waiting for an answer.

Wait Answer, similar to wait message, delays the process which uses it until an answer arrives in its queue.

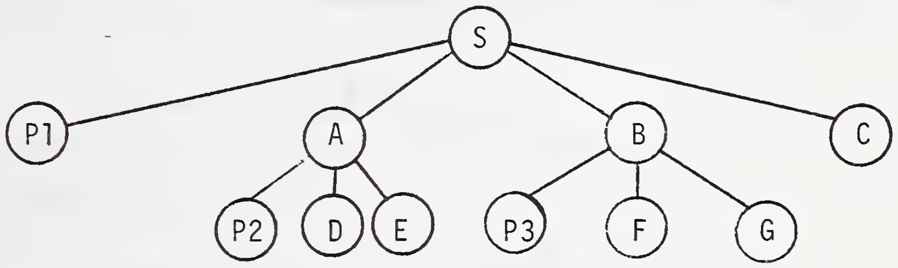
Synchronization between processes is obtained by using a sequence of the above primitives. The sequence Send Message, Wait Answer synchronizes the sending process with the receiving process which in turn, through the use of Wait Message, Send Answer is synchronized with the sending process.

As it was previously mentioned, the characteristics of the system- real time response, expandability, and common software feasibility- are dependent on the structure of the operating system. The hierarchical structure can be generalized to a tree structure which is quite suitable for achieving the above characteristics. For example:



In this example S, which is the system nucleus, creates processes A, B, and C. A and B in turn create D, E, F, and G. The operator of the system may have requested that the nucleus create processes A, B, and C, where process A assembles input messages, process B assembles output messages, and process C edits these messages. Process A creates processes D and E which are lines from the network and process B creates processes F and G where F is a local printer and G is a remote line.

The tree structure affords an improvement in scheduling processes. In some systems internal and external processes are scheduled differently whereby internal processes are assigned to one queue to which some fixed scheduling policy has been attached while external processes are assigned to other queues with possibly other fixed scheduling policies. A more flexible scheduling scheme is based on a tree structure system which has priority processes adjoined to various nodes. The priority processes may be used at each level of the tree to schedule other processes at that level. Since each priority process appears no different to the system than other processes, a priority process may be altered or replaced in the same manner as any other process. Thus the scheduling policy may not only be different at each subprocess, but the individual policies may be dynamically changed. In the preceeding example the priority process may be added as follows:



Process P1 then schedules A, B, and C. P2 schedules D and E, while P3 schedules F and G.

The concept of common software is more easily implemented in a system based on a tree structure. Here the various subprocesses, whether they are derivations of programs or input-output operations, can be linked to a main process as nodes are linked together on a tree. For example, suppose we have a network input operation which consists of assembling a message into a buffer, checking both for errors at the input character stream level and the assembled message level, and finally forwarding the assembled message to a teletype output operation. The input, which comes from a high speed line, is buffered into a channel type of device such as a multiplex or direct memory channel which enters a stream of bytes into a memory buffer. After a stream enters the buffer, it is checked for errors. The character stream is then transferred to another buffer in which the message is assembled. Later, the pointer to the assembled message is passed to the teletype output process which outputs the message character by character to a teletype. Finally the message buffer is released when the output has finished.

The common subprocesses of the above operations are:

Memory Management

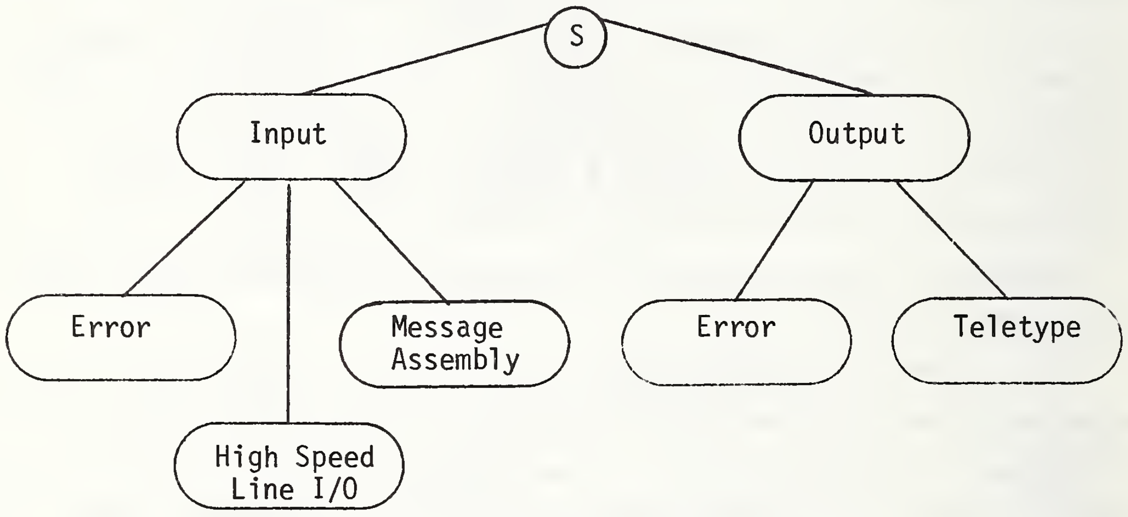
Message Assembly

High Speed Line I/O Routine

Error Routine

Teletype I/O Routine

The tree diagram of these processes is:



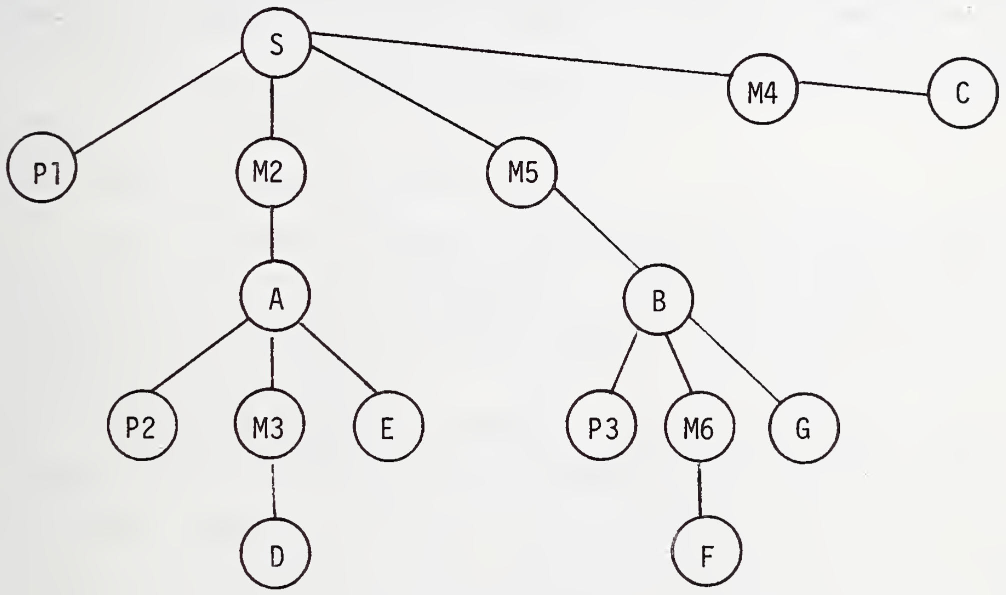
Other characteristics of the system are the following:

1. New operating system can be implemented as other programs without modification of the system nucleus.

2. Operating systems can be replaced dynamically thus allowing a communication system to switch among various modes of operation. Several operating systems can, in fact, be operating simultaneously.

3. Programs can be executed under different operating systems without modification, provided there is common agreement on the possible interprocess communication. If the interprocess communication is accomplished thru a convention like the communication primitives, the means of realizing that agreement is made easier.

4. System performance measurements may be made by placing measurement processes on the link to the process that is to be measured. Since the nucleus can create processes, such a placement can be made by making the measured process a subprocess of the measurement process. The following diagram illustrates this concept.



In this figure, M2, M3, etc., represent measurement processes that measure performance features of their respective subprocesses.

5.3 Other Feasibility Issues

The efficiency of common software will depend upon the potential for efficient processor storage management, in view of different frame, header, packet and message sizes, and also the possibility of some parameter-driven routines, common to all line disciplines handled. To assess these issues requires detailed commonality analysis, perhaps again using a finite state diagram technique, among the disciplines to be covered. Ultimately, a processor performance and resources projection would need to be made, for which a computer simulation would be advisable.

6.0 USER-ORIENTED NETWORK PROTOCOLS

User-oriented network protocols refer to software capabilities, uniformly implemented at all network computer sites, which reduce programming burden and otherwise assist users in exploiting the network resources. The term "protocol" signifies the concern with interactions between different computer systems. The impact of standard protocols for a network user is analogous to that of standard programming languages for any computer user - i.e., transferability of applications, repeatability of results, easier training, reduced software costs.

The needs and tradeoffs underlying the feasibility of standard protocols are examined in the following.

6.1 Overview of Types and Functions

From the user's terminal, the computers linked to a network may exist as three distinct environments: the local computer, the extended computer and the resource-shared computer, as distinguished by the following considerations.

The local computer, operating in a time-shared mode, permits the user to deal with his own files, either singly or as multifile sets. With permission from other users at the same site, he may extend his domain to make use of their files. When the user logs on the system from his terminal, the information which he must supply, insures the system that he is entitled to gain access to the limited environment defined under his identification and that all accounting will be so recorded. Using a formal command language, he can create, copy, move, modify, destroy, or use data stored in files under his domain. He may call upon subsystems available to all users, to perform more complex functions - such as an interactive text editor, or language translator. He may compile a source program, identified in his file, using a requested language translator, list the program on some device, file and execute it, with results assigned to a specified device. He may temporarily suspend the communication link to the computer, while the computer proceeds to perform a lengthy function, with the ability to reconnect to the process at some later time for the purpose of ascertaining process status or receiving results. An attempt is made in the time-shared system to human engineer the responses from the system and to prompt the user or display requested information, in order to ease the task of "thinking on line". If another user's file is requested to be accessed, information typed at the terminal is matched with information stored with the file, together with any security or protective information and verified before access may be made. When the user logs off at his terminal, all connections to the system are terminated, and information introduced or created and not previously filed is lost.

The extended computer environment enables the user to cause the local computer and another computer on the network to interact. To perform functions most efficiently, computers which interact should be compatible in organization and file structure. Fundamental to the operations of the extended computer concept, is the implementation of a network standard accounting system and security procedure. The primary functions visible to the user, not present in the local computer environment, are the Remote Job Entry function, and the File Transfer function. The Remote Job Entry function is a procedure whereby a user, at one Host computer, introduces a batch job to be run at some other Host computer on the network. The File Transfer function is a procedure for sending files from one Host computer and receiving the data in another Host as a new file, replacement file, or to augment an existing file. Because the name of a file which is being sent to another computer site may conflict with an existing file name in the receiving computer, provisions for changing the name are provided. In the extended computer concept, it is necessary for the user to supply the location and complete identification of files which are requested. This means that the user must know where the file resides in the network and what it is called in the remote computer, in order to use the file. When a user accesses many different computers, programs and data files over the network, the amount of specific detailed information which he must retain becomes significant.

The resource-shared computer can be made to appear as if there is no logical distinction between the resources that are local and those that are remote. Resource-sharing can make the services of other computers on the network available to a user without the user being concerned directly with the network details.

A computer network, built around a mission-oriented activity, can be applied to the sharing of data, programs, hardware-software systems, the services of people and facilities utilization, to the better achievement of the mission objectives. Where the hardware-software facilities on a network are semi-homogeneous, i.e., the same main frame even though different capabilities and services (eg. core size, I/O devices, specialized data files or programs) and varying workload, sharing of resources becomes a potential reality.

For any single activity, within the mission, responsibility of performing the: collection and verifying of new data, maintenance of a data file, warehousing of data files, computer program development, computer program testing and certification, computer program maintenance, program execution, and the printing of final results may each rest with a separate organization on the network, with additional back-up organizations for strategic reasons.

Organizations linked together over a computer network can become more responsive to change (eg., location of files, programs, and available facilities) provided the appearance of stability can be preserved for the user. Changes to a system can occur so fast, even in a network, that the written word cannot be dispensed to the affected user with adequate timeliness.

Programs performed on a routine basis involving input (eg., new data, program changes, etc.) from various organizations, require the user to know the identification and location of each file to be used with the set of programs, the hardware-software support requirements, availability of computer resources, as well as the sequence of control events which cause the process to be performed correctly. In an on-line batch oriented system, controls which govern the sequence of events in the computer process are either introduced by a sequence of control cards or by a sequence of control events which are stored within the system, such as catalogued procedures. The user at the terminal has the same need to preserve control sequences, so that this information need not be introduced at the terminal for each initiation of a job sequence.

To effect the appearance of stability over the network, for mission oriented activities involving more than a single computer site, it may be necessary to retain <u>at each site</u> specification files containing information relating to the current status of the system. These files would be maintained for each site by the mission office. In order to permit each computer site to be semi-autonomous in its file naming conventions, it would be necessary to superimpose, unique global names, controlled by the mission office, upon program and data file names in the specification files which are equated to the local name and computer location so that the effects of updating or modification of files by the responsible organization, would not be visible to the user who refers to the file by global name in the utility procedure described below. The four specification files are:

1. <u>Program Identification File</u>. For each program available on the network, its global name - equated to its local name for each location where the program resides - as well as the facilities required to perform the program, its security classification, organization responsible for its maintenance, date of issue, etc. would be stored in machine sensible form.

2. <u>Data Identification File</u>. For each data base available on the network, its global name - equated to its local name and location, and all pertinent information related to available use would be retained.

3. <u>Computer Resources File</u>. For each computer on the network, the list of resources, eg., core size, software support, I/O etc., would be retained.

4. Process Information File. This file contains a pseudo-job control stream for each application involving shared resources. The information would contain a sequence of events which constitute the order of performance of a stream of activities necessary to perform an entire sequence of programs with its data.

By the use of a utility program, operating at the applications level of the system, a specific process could be requested by name from the Process Information File. The utility program, using the pseudo-job control stream, would access the Program Identification file for the supplied global program name and determine (from the program resources required together with the Computer Resources File) which sites could supply the necessary resources. The data files identified by the control stream would be matched against the global names in the Data Identification File. The net effect would be to change all global names supplied in the control stream to the appropriate local name together with the site information and to create, where necessary, File Transfer commands. The resulting job stream would become the equivalent to a sequence of commands required to involve the File Transfer function and the Remote Job Entry function. If the workload status of each computer could be sent to all other computers on the network on a periodic basis automatically, then this information could be used to influence the distribution of the workload. This would permit control streams to be dynamically changed to reflect the situation on the network.

Such a facility would permit the mission office to maintain control over the location of information on the network and to balance the traffic and computer utilization, as well as change the responsibility of its organizations without effecting the user of the information. It would also permit the mission office to establish and maintain standards of program performance and formatting of data to the better utilization of the entire network. Any changes to the four files mentioned above would rest with the mission office upon receipt of information from the responsible organizations.

6.2 Software Implications of Standard Protocols

When a computer network is developed linking existing time-shared heterogeneous systems, the existing command language employed in the local environment cannot be changed readily to conform to some standard without the reprogramming of the system. The work now underway by the Command Languages Standardization Task Group of CODASYL and other pertinent work being done by NBS, could be applied in the future to assist in common definitions for various command functions. However, no practical degree of standardization of language is going to be complete in making heterogeneous systems appear to behave identically.

The feasibility of introducing a standard File Transfer function upon an existing time-shared system depends primarily upon the modularity of the current operating system and the ease of interfacing the function to the system software. Because this function does not exist in a local time-shared system, it should not become a hardship to conform to a standard, if the function can be imbedded at all. The current definition of the file transfer function (e.g., in ARPANET) requires the user to cope directly with many of the problems of moving data between non-homogeneous systems.

A standardized Remote Job Entry function for heterogeneous systems appears infeasible unless a standard job control language exists. Only the behavior, responses, and interaction with the File Transfer function can be defined. Because the usual job control language is based upon leading character (e.g., $ or //) recognition to alert the system to commands vs data, and no two systems use the same identical form, a consensus may not be obtainable at this time to implement a standard. Under the current definition of the Remote Job Entry function, it is necessary to know how each implementor performs the function. To implement a Remote Job Entry function on an existing system, could be a much more difficult task than a File Transfer function. How much of the function definition can be implemented is directly related to the current system capability and the ease with which the features could be interfaced.

6.3 A Specific Case: File Transfer Protocol

Two major considerations in the desirability of a protocol standard are: how well does it provide the user's needs? and, can it be implemented readily, with full conformity to specifications, by different installations, possibly with different hardware? Because of potential significance for the WWMCCS distributed data base planning, these questions were examined with regard to the File Transfer Protocol in ARPANET, a currently operational example of software for file-sharing. In particular, a very simple, ad-hoc application was conducted to ascertain whether this protocol could easily serve the most apparent needs, and whether any inconsistencies would be encountered in different implementations. The test provided information on the current status of FTP in ARPANET, and indicates the type of protocol evaluation which should be conducted, but on a more comprehensive, controlled basis.

6.3.1 Description of FTP:

The ARPANET File Transfer Protocol [6.1] is a uniform software capability to aid users to transfer files, programs, or data reliably and efficiently from one Host to another, thus allowing convenient use of the remote file storage and processing resources. It supports the following file operations:

Establish connections to the destination Host (Remote Host)
List remote directory (of files previously stored for the user)
Send local file (send, store, record name in remote directory)
Retrieve remote file (retrieve, store, record name in local directory)
Rename remote file
Delete remote file

The aim of FTP is to encourage the sharing of program and data files and the use of remote computers. It is intended to be implemented on computers of different word sizes, byte sizes, and internal representations, though not all Hosts have the specified FTP capabilities at the present time. A server may reject those byte sizes which it has not implemented, but must implement the default size of 8-bit bytes.

6.3.2 FTP Test in ARPANET:

The test was to verify that when program and data files are transferred from one Host to another, their integrity is maintained and that the results achieved in the execution of such a program at a remote Host were the same as those produced at a local Host. Because this test was being performed between homogeneous Hosts, the default values were used: i.e., a 36-bit connection and IMAGE type, were assumed. When transferring files between unlike systems, it may be desirable to convert characters into the standard NVT*-ASCII, with the conversions to and from the internal representations being performed by the receiving and sending Hosts. FTP provides a limited set of data type representations:

TENEX	for IMAGE, BYTE size 36
ASCII	for TYPE A, BYTE size 8
EBCDIC	for efficient transfer between Hosts which use EBCDIC internally
Image	recommended for the transfer of binary data

In the test, DEC PDP-10 computers under control of the TENEX interactive time-sharing system and File Transfer Protocol were used. Files were generated using the BASIC language, by way of a terminal connected to the NBS TIP. To make the verification, data and program files were generated at Host A (BBN-TENEX). The program was executed, the results noted (see figures 6-1 and 6-2), and the file names automatically entered into the directory.

*Network Virtual Terminal, as defined in the Telnet Protocol.

```
@BASIC

READY, FOR HELP TYPE HELP.
1,2,3,4,5,6,7,8,9
SAVE INDATZ

READY
NEW
NEW FILE NAME--PARKEZ

READY
10 FILES INDATZ
20 FILE #1,"INDATZ"
30 PRINT "X                    SQRT(X)"
40 INPUT #1,X
50 Y = SQRT(X)
60 PRINT X,Y
70 IF X < 10 GO TO 40
80 END
SAVE PARKEZ
```

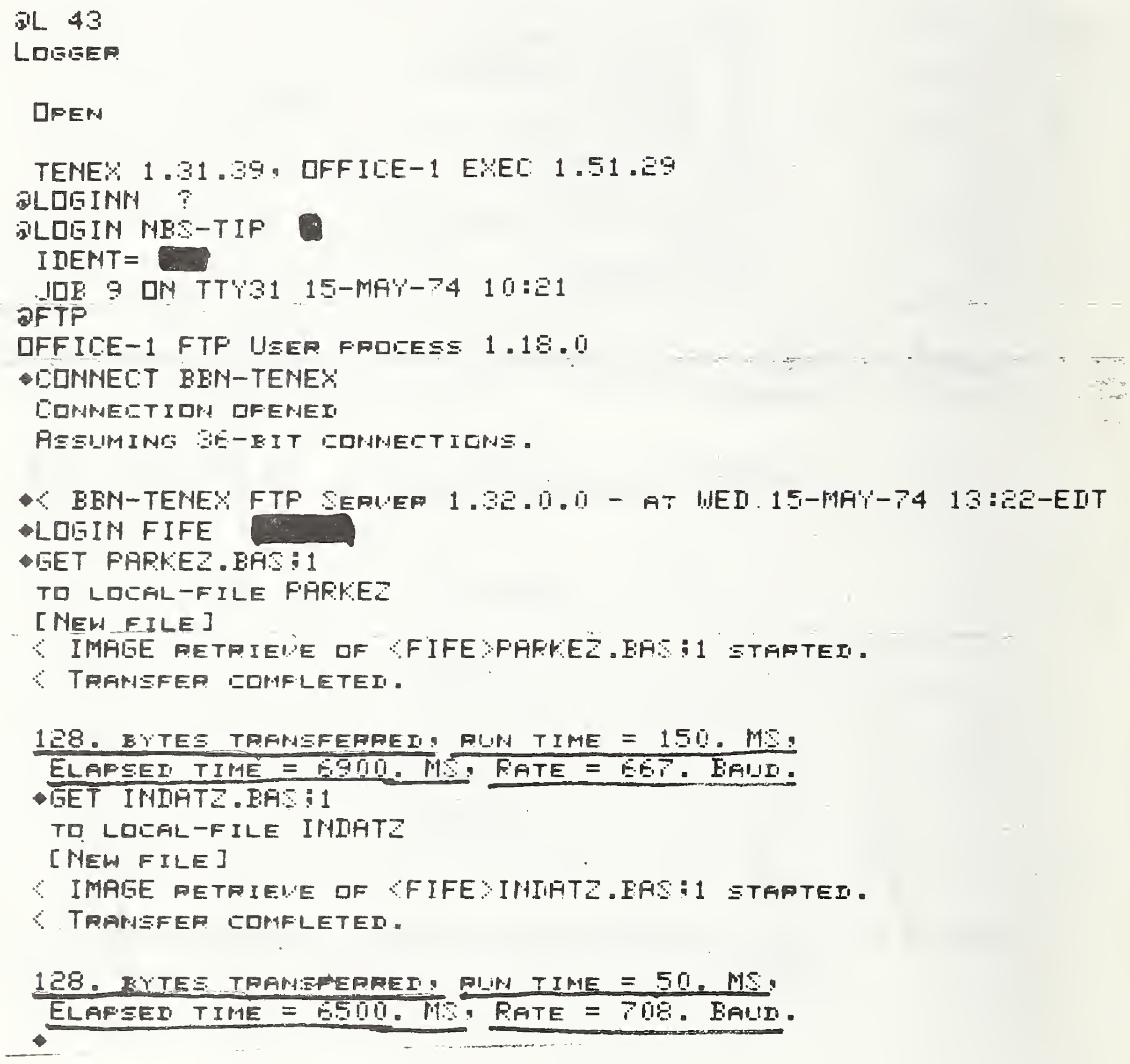
FIGURE 6-1. Source program and data created using BASIC programming language.

```
PARKEZ            13:19           15-MAY-74


X                 SQRT(X)
 1                 1
 2                 1.41421
 3                 1.73205
 4                 2
 5                 2.23607
 6                 2.44949
 7                 2.64575
 8                 2.82843
 9                 3
```

FIGURE 6-2. Results obtained from compilation and execution of source program.

The user then logged-out from Host A and logged-in at Host B (Office -1), called FTP and issued the CONNECT command for Host A. The connection was opened and the log-in procedure followed. A command to GET file name produced a dialogue between the user and the FTP in which the user was requested to give a name for the file and indicate whether it was a new file.



```
@L 43
Logger

 Open

 TENEX 1.31.39, OFFICE-1 EXEC 1.51.29
@LOGINN  ?
@LOGIN NBS-TIP [redacted]
 IDENT= [redacted]
 JOB 9 ON TTY31 15-MAY-74 10:21
@FTP
OFFICE-1 FTP User process 1.18.0
◆CONNECT BBN-TENEX
 Connection opened
 Assuming 36-bit connections.

◆< BBN-TENEX FTP Server 1.32.0.0 - at WED 15-MAY-74 13:22-EDT
◆LOGIN FIFE [redacted]
◆GET PARKEZ.BAS;1
 to local-file PARKEZ
 [New file]
 < IMAGE retrieve of <FIFE>PARKEZ.BAS;1 started.
 < Transfer completed.

 128. bytes transferred, run time = 150. ms,
  Elapsed time = 6900. ms, Rate = 667. Baud.
 ◆GET INDATZ.BAS;1
  to local-file INDATZ
  [New file]
 < IMAGE retrieve of <FIFE>INDATZ.BAS;1 started.
 < Transfer completed.

 128. bytes transferred, run time = 50. ms,
  Elapsed time = 6500. ms, Rate = 708. Baud.
 ◆
```

FIGURE 6-3. Execution of GET Command.

The FTP typed-out the file name, prefixing to it the directory name, when the retrieval of the file had started. Host A was then disconnected from Host B; a listing was produced of the new file which compared with that at Host A. Host C (USC-ISI) was then connected to Host B, the log-in procedure was completed, and the command to SEND file name was issued, the user was requested to confirm the file name and give a name to the new file. The file transfer then proceeded with the appropriate type-outs at the start and at the end. The user then disconnected Host C, logged-out from Host B, and logged into Host C.

```
@FTP
OFFICE-1 FTP User process 1.18.0
◆CONNECT USC-ISI
 Connection opened
 Assuming 36-bit connections.

◆< USC-ISI FTP Server 1.32.0.0 - at WED 15-MAY-74 10:29-PDT
◆LOGIN HUDSON
◆SEND PARKEZ.;1 [Confirm]

 to remote-file PARKEZ
< Store of <HUDSON>PARKEZ.;1;P777752;A3, Image type, started.
< Transfer completed.

128. bytes transferred, run time = 100. MS,
 Elapsed time = 25950. MS, Rate = 177. Baud.

◆SEND INDATZ.;1 [Confirm]

 to remote-file INDATZ
< Store of <HUDSON>INDATZ.;1;P777752;A3, Image type, started.
< Transfer completed.

128. bytes transferred, run time = 0. MS,
 Elapsed time = 14650. MS, Rate = 314. Baud.

◆DISCONNECT ◆USC-ISI
```

FIGURE 6-4. Execution of CONNECT, SEND and DISCONNECT Comands.

Files had then been created at Host A, transferred to Host B, and listed, then transferred from Host B to Host C.

At Host C, a DIRECTORY command showed the names of the files which had been transferred; a TYPE command to TENEX gave a listing of the program and a call to BASIC allowed the program to be identified and run. The results of the execution of the run at Host C compared as expected with those obtained at Host A. The FTP commands, whether issued at Host A, B, or C always produced the same results.

```
@BASIC

NEW OR OLD--OLD
OLD FILE NAME--PARKEZ

READY
RUN

PARKEZ          10:40          15-MAY-74



X               SQRT(X)
 1               1
 2               1.41421
 3               1.73205
 4               2
 5               2.23607
 6               2.44949
 7               2.64575
 8               2.82843
 9               3
```

FIGURE 6-5. Results obtained at Host C.

6.3.3 Status of FTP in ARPANET:

File Transfer Protocol, based on the latest information from the Network Information Center, has been implemented by approximately eight sites. In addition, two sites have other (private) file transfer protocols useable in restricted circumstances. A suitable user's guide to the use of FTP is not available and anyone desirous of using the system must glean enough information from the cryptic list of FTP commands and an implementor's specifications document.

6.4 PWIN Protocols

The PWIN project has already identified and begun implementation of several innovative protocols [6.2], that are consistent with resource-sharing concepts but without significant precedents. These include the Workload Sharing, Network Control Language, and Data Transfer Protocols. The available documentation and information within the scope of this project was not sufficient for thorough review and understanding, but clearly the restriction to a single system - the HIS 6000 with GCOS - should support a uniform implementation in WWMCCS. The notion of a "Directory Manager" described by Loret [2.1] corresponds in part to the needed information files described earlier in this section. Thus, from an overall standpoint, the area of user protocols still faces heavy development and operational evaluation before an effective approach emerges. Standardization at this time appears premature, but the importance of the area emphasizes the need for continued evaluation.

7. CONCLUSIONS FOR A STANDARDS DISCIPLINE

This report has undertaken an initial analysis of technical issues underlying networking standards for WWMCCS as it evolves from its present multi-media approach, experiencing progressive software and selective hardware upgrades, culminating possibly in a single standard interface with a DoD packet-switched network. Such envisioned development suggests concentration on three areas for standardization.

First, the adoption of standards for user-oriented network protocols is important in order to insulate operational users from a plethora of changes as the HIS 6000 and DN 355 software is upgraded. Such standards would also establish recognized, stable levels of resource-sharing capability, to ease the configuration management problem as seen by operational software users and developers. However, the degree of overall experience with resource-sharing protocols is modest at best, and considerable further development and experimentation must be expected before basic needs are well recognized and a consistent family of protocols emerges applicable to any computer network. For example, although the ARPANET is the forerunner, not all of its identified protocols have been uniformly implemented by major service sites, and no concerted efforts are underway in the community to fill some evident needs such as uniform access protocol and a network-wide software directory. Rather than criticism, this highlights the overall difficulty and cost in establishing standard user-oriented software to accommodate diverging applications interests with differing machines and operating systems. The progress that has been made within ARPANET on uniform file transfer capabilities for PDP-10/TENEX sites, which was briefly examined in this project, demonstrates that the needed protocols are not only feasible but can be effectively used in short order by the inexperienced user. The WWMCCS PWIN project includes an ambitious protocol development effort, partially based upon the ARPANET accomplishments and experience. An assessment of the currently conceived protocols as future WWMCCS standards can be initiated once the software is implemented and operational experiments have verified their functional sufficiency. In particular, a detailed study could be made of their dependence on HIS 6000/GCOS concepts, to yield recommended modifications to enhance transferability to other host systems. This was beyond the scope of this effort. In this regard, the current standards studies within CODASYL on operating system control language and at NBS on remote access protocols should be considered for pertinent results.

Second, the interim WWMCCS environment (before IDN) of multiple communications interfaces to the HIS 6000, suggests a feasibility study toward establishing a common interface processor, adaptable in capability for a mix of communications disciplines and variable traffic volume, and with one software approach incorporating all available communications control procedures. This concept would support a simplification of communications software within the host computers, and potentially would produce improved performance in communications handling. An examination of basic software design indicates that one conceptual organization would be feasible for scheduling and control of task-oriented software modules required to handle various disciplines concurrently. Issues of efficiency and performance in processor storage management and real-time control require further commonality analysis and simulation.

Third, the prospect that the new American National Standard ADCCP procedure would be used in the DoD, IDN, and the evident commercial interest in this procedure which would provide off-the-shelf communications terminals and equipment, is motivation for a comprehensive evaluation of its application in WWMCCS and packet-switched networks. A preliminary analysis of its complexity and information transfer performance relative to character-oriented bisynchronous procedures indicates ADCCP is at least competitive in basic practicality and efficiency. It appears, moreover, to have significant advantages for the implementation of higher order network controls and protocols. This has not been quantitatively evaluated, however, and is a suggested area for future standards analysis. NBS meanwhile is continuing its existing effort in development of national standards for performance evaluation of control procedures, and in measurement and analysis that will yield a comprehensive evaluation of ADCCP from the information transfer and error standpoint.

The work of this project, in an overall perspective, illustrates that standards analysis for computer networking is a challenging engineering activity, requiring a background in telecommunications as well as computer science or systems programming. The basic thrust in establishing a standards discipline is to undertake substantive evaluations, comparing the precision and clarity of alternative design specifications, establishing common functional characteristics, and deriving measures of performance, utility, and complexity.

REFERENCES

1.1 Systems and Software Division, "A Technical Guide to Computer-Communications Interface Standards", Technical Note 843, National Bureau of Standards, August 1974.

2.1 B. J. Loret, "Prototype World-Wide Military Command and Control System Intercomputer Network", AIAA Conference Paper, JTSA, 1974.

2.2 Defense Communications Agency, _Annex A to Subsystem/Project Plan 2-74_, 1974.

3.1 Martin, James. T., _Systems Analysis for Data Transmission_, Englewood Cliffs, New Jersey, Prentice-Hall, 1972.

3.2 Becker, Hal B., _Functional Analysis of Information Networks_, New York, John Wiley & Sons, 1973.

3.3 PRC Data Services Company, _WWMCCS Common Communications Executive, Functional Definition_, Prepared for Defense Communications Agency, December 2, 1973.

3.4 Honeywell Information Systems, Inc., _Series 6000 Summary Description_, Manual DA48, 1971.

3.5 Heart, Frank E., et. al. "The Interface Message Processor for the ARPA Computer Network", _Spring Joint Computer Conference_, 1970, pp. 551-567.

3.6 Carr, C. Stephen, Stephen D. Crocker and Vinton G Cerf, "Host-Host Communication Protocol in the ARPA Network", _Spring Joint Computer Conference_, 1970, pp. 589-597.

3.7 Crocker, Stephen D., et. al. "Function-Oriented Protocols for the ARPA Computer Network", _Spring Joint Computer Conference_, 1972, pp. 271-280.

3.8 Koehr, G. J., "A Summary Description of WWMCCS Hardware and Software", Mitre Corporation, Working Paper No. 4202, February 1972.

4.1 Stutzman, Bryson W., "Data Communication Control Procedures", Computing Surveys, Vol 4, No 4, Dec. 1972, pp 197-220.

4.2 ANSI X3.28-1971, American National Standard Procedures for the Use of the Communication Control Characters of American National Standard Code for Information Interchange in Specified Data Communications Links, American National Standards Institute, Inc., Mar. 10,11971.

4.3 ISO/TC 97/SC 6 (Stockholm - 9), High Level Data Link Control Procedures Proposed Draft International Standard on Command and Responses (HDLC), June 1973.

4.4 ANSI X3S3.4/475 DR7 - Proposed American National Standard on Advanced Data Communication Control Procedures (ADCCP), December 21, 1973.

4.5 Bjorner, Dines, "Finite State Automaton - Definition of Data Communication Line Control Procedures", Fall Joint Computer Conference, 1970.

4.6 Nakajo, Toshihiko and Nagata, Tetsuo, "On the Packet Interleaved Interface Between Packet Switched Network,and Computers", Datacom 73, pp 104-112.

4.7 Bolt, Beranek and Newman, Specifications for the Interconnection of a Host and an IMP, (April 1972 Revision), Report No. 1822.

4.8 Bolt, Beranek and Newman, Inc., Technical Information Report No. 89, "The Interface Message Processor Program", Nov. 1973.

4.9 Prototype WWMCCS Intercomputer Network, Communications Software Programming Specification, DCA N341, March 16, 1973.

4.10 ANSI X3S3.5/80, Proposed American National Standard - Determination of Performance of Data Communication Systems, Sept. 30, 1971.

4.11 Determination of Performance of Digital Data Communication Systems, R. T. Moore, International Conference on Communications, 1971 Proceedings, Institute of Electrical and Electronic Engineers, Inc., New York, New York, pp 12-1 - 12-3.

4.12 NBS Technical Note 779,"Data Communication System Throughput Performance Using High Speed Terminals on the Dial Telephone Network," Dana S. Grubb, May 1973.

5.1 Dijkstra, W., "The Structure of the "THE"-Multiprogramming System", Comm. ACM, Vol 11, No 5, May 1968, pp. 341-346.

5.2 Hansen, Brinch P., "A Comparison of Two Synchronizing Concepts", Acta Informatica, Vol 1, No 3, 1972, pp. 190-199

5.3 Hansen, Brinch P., "The Nucleus of a Multiprogramming System", Comm. ACM, Vol 13, No 4, April 1970, pp. 238-250.

5.4 Dijkstra, E. W., "Hierarchical Ordering of Sequential Processes", Acta Informatica, Vol 1, No 2, 1971, pp. 115-138.

5.5 Varney, R. C. and M. H. Gotterer, "The Structural Foundation for an Operating System," The Computer Journal, Vol 16, No 4, November 1974, pp. 367-369.

6.1 N. Neigus, File Transfer Protocol, ARPANET Network Working Group NIC #17759 12 July 1973.

6.2 Computer Sciences Corporation, Network Control Language, Data Transfer Protocol and Workload Sharing Functional Specifications, June 1, 1973.

NBS-114A (REV. 7-73)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. *NBS IR 74-570*	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE *STANDARDS ANALYSIS FOR FUTURE WWMCCS COMPUTER NETWORKING*	5. Publication Date *August 30, 1974*
	6. Performing Organization Code
7. AUTHOR(S) *DENNIS W. FIFE, Editor*	8. Performing Organ. Report No.
9. PERFORMING ORGANIZATION NAME AND ADDRESS NATIONAL BUREAU OF STANDARDS DEPARTMENT OF COMMERCE WASHINGTON, D.C. 20234	10. Project/Task/Work Unit No. *640.1400*
	11. Contract/Grant No.
12. Sponsoring Organization Name and Complete Address *(Street, City, State, ZIP)* *Joint Technical Support Activity Defense Communications Agency Washington, D. C. 20305*	13. Type of Report & Period Covered *Final*
	14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

The World-Wide Military Command and Control System (WWMCCS) comprises over 30 computer installations, evolving toward heavy dependence on terminal-to-computer and computer-to-computer communications in serving individual bases, commands, and National military authorities. Although a highly standardized system has emerged through identical mainframes, hardware devices, and basic software, the hardware/software configurations of individual installations are not identical and significant evolutionary changes are occurring in hardware/software components, interfaces, and configuration rules. It is vital therefore to deliberately adopt selected interface specifications as standards that transcend the present computer hardware and software, and that will guide near-term reconfiguration or redesign as well as ultimate, progressive replacement of the WWMCCS computer network system. A previous NBS report, Technical Note 843, provided a succinct handbook on existing computer-communications standards with widespread applicability. This report addresses issues and methods in interface standards analysis, to identify and partially evaluate computer-communications techniques that would contribute to enhancing WWMCCS interoperability. These issues include: functional decomposition of computer-communications processes; complexity and performance analysis of advanced data link control techniques; feasibility of common communications front-end software; and requirements for user-oriented network protocols. The presentation indicates the standard "discipline" involved in the evaluation and implementation of effective standards for a complex computer network environment.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)*

Command and control, communications performance analysis, computer communications, computer networks, computer standards, digital communications, World-Wide Military Command and Control System (WWMCCS).

18. AVAILABILITY ☐ Unlimited	19. SECURITY CLASS (THIS REPORT) UNCLASSIFIED	21. NO. OF PAGES 107
[✓ XX] For Official Distribution. Do Not Release to NTIS		
☐ Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Cat. No. C13	20. SECURITY CLASS (THIS PAGE) UNCLASSIFIED	22. Price
☐ Order From National Technical Information Service (NTIS) Springfield, Virginia 22151		